EmbeddedBlue™ 506

User Manual

Part Number 0000100 – Revision A

Last revised on February 18, 2005 – Printed in the United States of America

A7 Engineering, Inc.
12860 C Danielson Court
Poway, CA 92064

Table of Contents

Table of Figures

Table of Tables

Introduction

Congratulations on your purchase of the EmbeddedBlue 506 (eb506) serial Bluetooth module. The eb506 is an add-on component to the Z-World RCM3000, RCM3100, RCM3200, and RCM3300; enabling wireless communications with other Bluetooth devices including cellular phones, handheld computers, PCs, and other serial port adapters. Hobbyists, developers, and OEMs can take advantage of advanced wireless connectivity with this easy to use module.

The eb506 module provides a point to point connection much like a standard serial cable. Connections are made dynamically and can be established between two eb506 modules or an eb506 module and a standard Bluetooth v1.1 or v1.2 device. Devices can be dynamically discovered and connected in an ad-hoc manner.

Manual Conventions

Below is a list of typographical conventions used in this manual:

`Text in this font`

- `Is used to show data that is sent to the eb506.`
- `Inside a gray box is used to show data that is sent from the eb506.`

`Text in this font`

- `Is used to show source code`

In the command set section of this manual

- Required parameters and placeholders appear in standard lowercase type.
- Placeholders appear in *italics*. For example, if *address* shows up in a syntax line, the actual address of the device must be entered.
- Required parameter options are separated by a vertical bar |.
- Optional parameters are enclosed in brackets [ ].

Getting More Information

The Bluetooth website, www.bluetooth.com, contains the Bluetooth specification, profiles, and other documents relevant to Bluetooth.

General information regarding the eb506 module, EmbeddedBlue, and other Bluetooth products from A7 Engineering can be found on the A7 website at www.a7eng.com.

Technical support for EmbeddedBlue products is available via email support@a7eng.com and an online discussion forum www.a7eng.com/support/forum/forum.htm. It is recommended that you make use of the online forum, as many questions may be answered quickly by others, and you may even find the answer to your question already in a post.

A7 Engineering also provides professional design services on a contract basis to anyone requiring assistance with their design and/or development of Bluetooth products. For further information visit the A7 Engineering website www.a7eng.com/services/services.htm.

Bluetooth Overview

What is Bluetooth?

To put it simply, Bluetooth is a technology standard for electronic devices to communicate with each other using short-range radio. It is often referred to as a “cable replacement” technology, because it is commonly used to connect things, such as cameras, headsets, and mobile phones that have traditionally been connected by wires. Bluetooth is much more than simply a way to cut the cord between today’s existing electronic devices. It is an enabling technology that will take these devices to new levels of productivity and functionality and enable a whole new class of devices designed with communications and connectivity in mind.

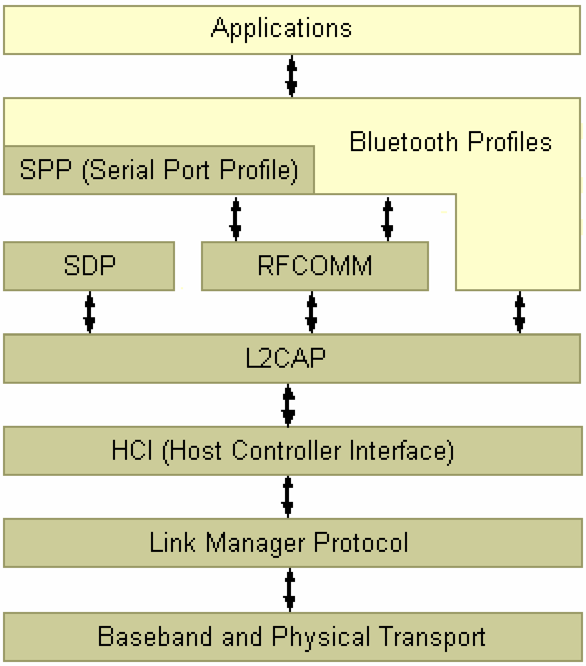
The Bluetooth Special Interest Group (SIG) defines Bluetooth a bit more broadly as the "worldwide specification for small-form-factor, low-cost radio solutions that provide links between mobile computers, mobile phones, other portable devices, and connectivity to the Internet." In defining Bluetooth, the SIG has taken a very different approach than the IEEE 802.11 Committees did. Rather than build Bluetooth as an adjunct to TCP/IP, it was defined as a standalone protocol stack that includes all layers required by an application. This means that it encompasses not only wireless communications but also service advertisement, addressing, routing, and a number of application-level interfaces referred to as profiles.

Bluetooth is based on a frequency hopping spread spectrum (FHSS) modulation technique. The term spread spectrum describes a number of methods for spreading a radio signal over multiple frequencies, either simultaneously (direct sequence) or in series (frequency hopping.) Wi-Fi devices are based on direct sequence spread spectrum transmission which uses multiple channels simultaneously. While this technique increases the speed of transmission (for example in Wi-Fi from 1.5MHz to 11MHz), it is more susceptible to interference from other radio sources as well as being a greater source of interference to the surrounding area.

In contrast, Bluetooth utilizes the frequency hopping method of spread spectrum which uses multiple radio channels to reduce interference and increase security. The signal is rapidly switched from channel to channel many times per second in a pseudo-random pattern that is known by both the sender and receiver(s). This provides robust recovery of packet errors caused by interference from another radio source at a particular frequency. Also, data is generally more secure because it is not possible to receive more than a fraction of the data unless the hopping pattern is known. Bluetooth utilizes frequency hopping in the 2.4GHz radio band and hops at a relatively fast pace with a raw data rate of about 1 Mbps. This translates to about 700 kbps of actual useful data transfer. The eb506 module supports a maximum sustained bidirectional data rate of 230.4kbps.

What is a Profile?

Bluetooth devices can support interoperability with one or more types of devices. In order for two Bluetooth devices to communicate with each other, they must share at least one common profile. If I want a Pocket PC to communicate with my EmbeddedBlue radio I need to make sure that they both support the same profile. EmbeddedBlue devices support the Serial Port Profile (SPP) which is one of the earliest and most widely supported profiles.


Applications
Bluetooth Profiles
SPP (Serial Port Profile)
SDP
RFCOMM
L2CAP
HCI (Host Controller Interface)
Link Manager Protocol
Baseband and Physical Transport

The main elements of the Bluetooth stack are shown in the figure to the right. As with a typical diagram of the TCP/IP stack, there are a number of details that are hidden by the apparent simplicity of the stack. Specifically, there are a number of profiles that sit roughly on top of the L2CAP layer that provide much of the power (and also the complexity) of the Bluetooth protocols.

These profiles are the primary entry into the stack for an application. Essentially, they define the set of services that are available to that application. Currently there are more than 25 different profiles defined or in the process of being defined by the Bluetooth SIG. With so much variety, acquiring an in-depth understanding of Bluetooth is not a trivial task. However, the abstraction by a single profile can provide an application the use of the profile without such detailed knowledge.

There are a number of profiles that are exposed in very familiar forms. The eb506 module, for instance, implements the SPP profile which enables it to appear like a traditional serial port. This virtually eliminates the need for the user to have specific Bluetooth knowledge and allows the radios to be integrated into applications very quickly.

Bluetooth and Wi-Fi

Bluetooth and Wi-Fi are often compared to each other because they are both capable of providing networking on the 2.4GHz consumer frequency band. Many of the differences between these two technologies can be traced to the fact that networking was not the primary design goal for Bluetooth as it was for Wi-Fi. With a greater transmission range (about 100 meters indoors) and larger bandwidth (about 11Mbps), Wi-Fi is typically the better choice for wireless LANs and Internet connectivity.

Bluetooth on the other hand was designed for driverless, cordless connectivity between devices. Because Bluetooth transmitters are smaller in size, have lower power demands, a more limited range (10 - 100 meters) and narrow bandwidth (1Mbps), they are better suited for use in embedded and mobile devices that exchange smaller amounts of information while conserving power and space.

While their functionality does not compete directly, 802.11b and Bluetooth do compete for the airwaves. Since they both operate on the 2.4GHz band of the ISM radio spectrum, these two wireless technologies may interfere with each other. Bluetooth devices minimize interference by employing a frequency-hopping spread spectrum scheme that changes the frequency used about 1600 times per second. Unfortunately, since Wi-Fi uses a direct sequence spread spectrum method, this also means that Bluetooth transmissions will collide with those of any nearby 802.11b devices and slow Wi-Fi data transmission rates. The Bluetooth SIG and its member companies have put a lot of effort into coexistence solutions for these two standards and are very committed to ensuring that these devices work well together.

While 802.11b was designed solely for data communications, Bluetooth takes things quite a bit further. A key component of the Bluetooth standard is its notification and service discovery mechanism. This allows Bluetooth devices to identify themselves and describe their capabilities to other Bluetooth devices in the area. For instance, the Dial-Up Networking profile defines how discoverability can be used to locate and connect to other devices such as a cellular phone that supports the same profile. The profile then describes how to dial the phone, connect to either analog or data services, and control the connection seamlessly. This combination of dynamic discovery of services and built in definitions of the services goes well beyond anything offered by the 802.11b protocol.

Security

Bluetooth security is defined by three main elements: availability, access, and confidentiality. It is important to distinguish between these elements because Bluetooth security is also highly configurable so that it can meet the needs of devices in many different scenarios. An understanding of the basics will provide the knowledge that you need to choose a security strategy for your device.

The first important element of Bluetooth security is availability. If a device cannot be seen or connected with, it is obviously quite secure. Bluetooth defines both of these features as part of the security model and they are exposed by the EmbeddedBlue device through the *set visible* and *set connectable* commands. This is a very coarse level of control, but it is also quite effective and can be used in combination with other security features.

The second and most complex element of Bluetooth security is access control. This type of security is only relevant when the module is *connectable* and is designed to provide protection in this case. The general idea is that remote devices must become trusted before they will be allowed to connect and communicate with the EmbeddedBlue module. In order to become trusted, a remote device must present a passkey that matches the stored local passkey. This only needs to be done once, as both devices will remember their trusted status and allow future connections with that specific device without exchanging passkeys again.

The EmbeddedBlue module uses the *set security* command to configure access control. There are three possible settings for security, *off*, *open*, and *closed*. When security is turned *off*, connection attempts will be allowed from all remote devices. When security is set to *open,* connections are only allowed from trusted devices, but new devices can become trusted by presenting the correct passkey. Forming a trusted relationship is carried out automatically in this mode the first time that a remote device connects with the EmbeddedBlue module. When security is set to *closed*, only connections from trusted devices will be allowed and no new devices may become trusted. Closed security is the most restrictive setting and therefore the most secure.

The last element of Bluetooth security is confidentiality. Once a link with a trusted device has been established, it may be important to know that the data being transmitted cannot be intercepted by a third party. All transmitted data can be encrypted by configuring the *encrypt* setting to *on*. This only has an effect when security is set to either *open* or *closed*. The EmbeddedBlue module supports 56-bit encryption by default, but 128-bit encryption is available. Due to export restrictions to certain countries, firmware supporting 128-bit encryption is only available with proper approval from A7 Engineering.

The Basics

Most of the complexity of working with Bluetooth has been encapsulated in the EmbeddedBlue module in order to make it easier to use. The specific application profile that is supported is SPP, or the Serial Port Profile. This is the most popular and convenient protocol for many embedded applications of Bluetooth since it emulates a simple serial port link between devices. Once the connection is set up, it is a simple matter to communicate between the endpoints of that connection using familiar and well-supported programming constructs as will be shown by the numerous programming examples throughout this manual.

The A7 Engineering Helper Library

As you will see in the samples throughout this manual, a typical application that uses an eb506 module needs to do a bit of string parsing to determine when commands have completed or retrieve command results. To simplify this for you, A7 has provided a helper library named A7Eng_eb506.lib. Before running the samples that are dependant on this library you will need to install it into the Dynamic C library path using the following instructions.

Step 1: Install the Library

In this step we will install the A7 Engineering library and make it available to all Dynamic C applications.

1. Insert the EmbeddedBlue **CD** that came with the eb506 module into your **PC**.
2. Copy the file **\Samples\Dynamic C Applications\A7EngLib\A7Eng_eb506.lib** from the CD into the Dynamic C Library folder **\DCRABBIT_9.01\Lib\A7** on your PC.
3. Open the file **\DCRABBIT_9.01\LIB.DIR** in notepad so that it can be edited.
4. **Add** the following **text** to the very bottom of the file

   LIB\A7\A7ENG_eb506.LIB
5. Click **Save** from the File menu to save the changes.

Now that the helper library is installed into the Dynamic C library path it can be used by any Dynamic C application by adding the following text as the first line of code.

```
#use a7eng_eb506.lib
```

Each of the code samples in this manual illustrate the use of the helper library as well as the functions that it contains.

Pass-Through Prototyping

One of the best ways to learn about anything new is through hands-on experimentation. To make this easy with the eb506, the following sample will enable you to communicate directly from your PC to the radio module. The Rabbit Core Module (RCM) becomes a simple pass-through moving data between the serial port on the PC and the eb506.

To perform this exercise, as documented, you will need one of the supported RabbitCore modules, one Prototyping board, one eb506 module, and a PC with a serial port.

Step 1: Insert the eb506 Module into the RabbitCore and Prototyping Board

In this step we will insert the eb506 module into the RabbitCore and Prototyping board.

Note about serial ports: There are two jumpers (J1 and J2) on the eb506 module which allow the user to have control over which serial port to use for transmitting and receiving data. The eb506 module allows you to use either port B or port F. The samples in this manual use port F for both Rx and Tx. If you are already using port F, then switch the jumper to port B and substitute the correct serial methods in your program, ex: use SerBopen() instead of SerFopen().

CAUTION: Care should be used in the following steps to insure that the boards are connected with the correct orientation, using the mounting hole as a guide, to prevent damage.

1. Remove **power** from the **Prototyping** board.
2. Remove the **RabbitCore** module from the **Prototyping** board.
3. Using the **mounting hole** as a guide, insert the compatible **RabbitCore** module into the **eb506 module**, ensuring that J2 of the RabbitCore module is inserted into connector CN4 of the eb506 module.
4. Using the **mounting hole** as a guide, insert an **eb506** module, with RabbitCore attached, into the **Master Module Connectors** of the Prototyping board, assuring that CN1 of the eb506 module is inserted into connector JA of the Prototyping board. Ensure that the eb506 is inserted into the area marked MASTER.

Step 2: Write a Dynamic C Application for Pass-Through Prototyping

In this step we will write a Dynamic C application to perform pass-through prototyping between the eb506 radio module and a PC.

1. Open the **Dynamic C Editor**.
2. Enter the following **program code** into the editor. This application is available in electronic form on the accompanying CD, in the Samples folder, in the file Passthrough.c.

```
#use a7eng_eb506.lib
#class auto

void main()
{
   auto char szPortAData[32], szPortFData[32];
   auto int nDataOnA, nDataOnF, nDataReadFromA, nDataReadFromF;

    // Open the serial port for communications and initialize the eb506
   serAopen(9600);
   serFopen(9600);
   a7_serFeb506Init();

   while(1)
   {
      BigLoopTop();

      // Read data from the diagnostics port and send it to the eb506
      costate
      {
         nDataOnA = serArdUsed();
         if(nDataOnA > 0)
         {
            wfd nDataReadFromA = cof_serAread(szPortAData, nDataOnA, 50);
            wfd cof_serFwrite(szPortAData, nDataReadFromA);
         }
      }
```

```
            // Read data from the eb506 and send it to the diagnostics port
            costate
            {
               nDataOnF = serFrdUsed();
               if(nDataOnF > 0)
               {
                  wfd nDataReadFromF = cof_serFread(szPortFData, nDataOnF, 50);
                  wfd cof_serAwrite(szPortFData, nDataReadFromF);
              }
            }
         }
      }
```

3. On the **File** menu, click **Save As**.
4. In the **File name** box, enter a file name to which to save the program just created. For example, Passthrough.c.
5. Click **Save**.

Step 3: Download the Application to the RabbitCore Module

In this step we will download the application that we just created to the RabbitCore board.

1. Select the **Passthrough.c** application in the Dynamic C Editor.
2. Connect the **RabbitCore Module** to the **PC** via the serial cable.
3. Apply **power** to the ZWorld **Prototyping board**.
4. On the **Compile** menu, click **Compile**.

   The application will be downloaded to the RabbitCore module and stored in flash. For this sample and all others, ensure the correct project options have been configured for your setup. Choose “Project Options” from the Options menu. Select the correct serial port as well as other serial options.
5. Switch the **serial** cable connection on the RabbitCore module from the programming (**PROG**) header to the diagnostics (**DIAG**) header.

   Note about debugging: You might see an error saying “Timeout while waiting for response from target.” This is due to the RabbitCore module’s use of serial A for debugging, which we are using for diagnostics in this sample. This means that you cannot step through code with this sample.

Step 4: HyperTerminal Setup

In this step we will setup the Windows HyperTerminal application to establish a connection with the RabbitCore module.

1. Open **HyperTerminal**.

   This will display the Connection Description dialog.

2. In the **Name** box, type the name of your connection. (e.g. eb506).
3. Click **OK**.

   This will display the Connect To dialog.

4. In the **Connect using** dropdown, select the **serial port** to which you have connected the RabbitCore Module.
5. Click **OK**.

   This will display the properties dialog.

6. In the **Bits per second** dropdown, select **9600**.
7. In the **Data bits** dropdown, select **8**.
8. In the **Parity** dropdown, select **None**.
9. In the **Stop bits** dropdown, select **1**.
10. In the **Flow control** dropdown, select **None**.
11. Click **OK**.

    This will establish a connection to the serial port assuming that no other devices are using the serial port. If another device is using the serial port, disconnect the other device using the associated application or choose a different serial port to connect the eb506.

12. On the **Call** menu, click **Disconnect**.

    This will disconnect the connection just established so that we can modify the connection properties as follows.

13. On the **File** menu, click **Properties**.

    This will display the properties dialog.

14. On the **Settings** tab, click **ASCII Setup**.

This will display the ASCII Setup dialog.

15. Check the **Send line ends with line feeds** checkbox.
16. Check the **Echo typed characters locally** checkbox.
17. Check the **Append line feeds to incoming line ends** checkbox.
18. Check the **Wrap lines that exceed terminal width** checkbox.
19. Click **OK**.

    This will return to the properties dialog.
20. Click **OK**.
21. On the **Call** menu, click **Call**.

    This will establish a connection with the serial port.
22. **Reset** the RCM Prototyping board.

    This forces the program we just wrote to begin execution.
23. Press the **“Enter”** key to send a carriage return to the eb506 module.

    You should see a caret appear in HyperTerminal; this is the prompt for the eb506.

Step 5: Execute a few eb506 commands

In this step we use the connection that has been established to execute a few simple eb506 commands.

1. Using HyperTerminal, get the **version** of the **eb506** module by using the Version command.

   Example:

```
>ver all

ACK

Firmware Version: 2.0

Firmware Build: 247

Model Number: eb506

Serial Number: 238

Manufacturer: A7 Engineering

>
```

2. Using HyperTerminal, get the **address** of the **eb506** module by using the Get command.

   Example:

   ```
   >get address
   ACK
   00:0C:84:00:05:29
   >
   ```

   Experiment with other commands simply by typing them in to the HyperTerminal window. The HLP command is a great place to start.

Command Mode

The eb506 supports two main operating modes: command mode and data mode. Upon power up, the eb506 enters command mode and is ready to accept serial commands. The factory default communication parameters are 9600 Baud, 8 Data Bits, 1 Stop Bit, No Parity, and No Flow Control. The eb506 supports commands to modify the baud rate and flow control settings.

In this mode there are a number of commands that can be sent to change the baud rate, locate other devices that are in range, check the firmware version, etc. All commands are sent using visible ASCII characters (123 is 3 bytes “123”). Upon the successful transmission of a command, the ACK string will be returned. If there is a problem in the syntax of the transmission then a NAK string is returned. After either the ACK or NAK, a carriage-return <CR> character is returned. When a prompt (<CR> followed by a ‘>’) is returned, it means that the eb506 radio is in the idle state and is waiting for another command. White space is used to separate arguments of the command and a carriage-return <CR> (ASCII 13) is used to mark the end of the command.

Data Mode

Once the eb506 radio is connected to another Bluetooth device, the eb506 automatically switches into data mode. All data transmitted while in this mode will be sent to the remote device and, therefore, NO further commands can be sent until the eb506 radio is disconnected or switched back to command mode by use of the mode control I/O line or the Switch to Command Mode sequence.

The connection status line of the eb506 module can be monitored to determine if there is an active connection. Additionally, whenever a connection is present, the Connection Status LED on the eb506 module will be on.

I/O Lines

The eb506 module features two 34 pin headers for connecting to the supported ZWorld RabbitCore modules. A full device pin out is available in the Technical Specifications section of this manual. There are several pins that are important when performing the exercises in the Establishing a Connection and the Communications sections of this manual.

> Pin 13 on connector CN2 of the eb506 module, which aligns with the pin designated "PE7" on the RabbitCore module, is the On\Off pin. A RabbitCore module application can drive this pin low to power the module down or high to power the module up.
>
> Pin 27 on connector CN1 of the eb506 module, which aligns with the pin designated "PD4" on the RabbitCore module, is the Connection Status pin. A RabbitCore module application can interrogate this pin to determine the connection status of the eb506 radio.
>
> Pin 28 on connector CN1 of the eb506 module, which aligns with the pin designated "PD5" on the RabbitCore module, is the Mode Control pin. A RabbitCore module application can drive this pin high to enter Data Mode or low to enter Command Mode.

Resetting the eb506 to the Factory Default Settings

There are two different mechanisms to reset the eb506 module to the factory default settings. Either by shorting the STATUS and MODE pins (PD4 and PD5) and then applying power to the eb506 module, or by issuing the reset command to the eb506; see the command set reference at the back of this manual for the syntax of the reset command.

Switching between Data Mode and Command Mode

When a Connection command is issued, the eb506 attempts to establish a connection to the device with the address specified in the command. Once a connection is established, the eb506 switches into data mode. At this point all data sent to the eb506 is transmitted to the remote Bluetooth device over the wireless link. It is possible to switch from data mode to command mode, issue commands, and then return to data mode, while maintaining a connection. The eb506 allows you to switch between data mode and command mode by issuing the Switch to Command Mode and Return to Data Mode commands or by driving the MODE control I/O line (PD5) of the eb506 module.

The following Dynamic C application uses the Switch to Command Mode and Return to Data Mode serial commands to switch between data mode and command mode. This application is available in electronic form on the accompanying CD, in the Samples folder, in the file CmdModeSoft.c.

```
  auto char szPortFData[32];
```

```
memset(szPortFData, 0, sizeof(szPortFData));

// Open the serial port for communications and initialize the eb506
serFopen(9600);
a7_serFeb506Init();

// Connect to the remote device
serFputs("con 00:0C:84:00:05:29\r");
a7_serFwaitForString("ACK\r", 0);

// Wait for the connection to be established
while(!BitRdPortI(PDDR, 4));
printf("Connection established\n");

serFputs("This string is sent in data mode\r");

// Switch to Command Mode
a7_pauseMs(2000);
serFputs("+++");
a7_serFwaitForString("\r>", 0);

printf("In Command Mode\n");

// Get local eb506 Bluetooth Address
serFputs("get address\r");
a7_serFwaitForString("ACK\r", 0);

// Read local address from get command
while(serFpeek() == -1) {}
serFread(szPortFData, 17, 10);
szPortFData[17] = '\r';
printf("Local eb506 address: %s\n", szPortFData);
a7_serFwaitForString("\r>", 0);

// Return to data mode
```

```
serFputs("ret\r");
a7_serFwaitForString("\r>", 0);

// Send data through eb506
serFputs("My eb506 address is ");
serFputs(szPortFData);

// Switch to Command Mode
a7_pauseMs(2000);
serFputs("+++");
a7_serFwaitForString("\r>", 0);

printf("In Command Mode\n");

// Disconnect from remote device
serFputs("dis\r");
a7_serFwaitForString("\r>", 0);

printf("Disconnected\n");

serFclose();
```

The following Dynamic C application uses the mode control I/O line of the eb506 module to switch between data mode and command mode. Switching between data mode and command mode via the mode control I/O line is often preferred, as it is faster than the serial method. This application is available in electronic form on the accompanying CD, in the Samples folder, in the file CmdModeHard.c.

```
auto char szPortFData[32];
memset(szPortFData, 0, sizeof(szPortFData));

// Open the serial port for communications and initialize the eb506
serFopen(9600);
a7_serFeb506Init();

// Connect to the remote device
serFputs("con 00:0C:84:00:05:29\r");
```

```
a7_serFwaitForString("ACK\r", 0);

// Wait for the connection to be established and switch into data mode.
// When switching into data mode, a 50ms timeout is required to give the
// module enough time to make the change.
while(!BitRdPortI(PDDR, 4));
BitWrPortI(PDDR, &PDDRShadow, 1, 5);
a7_pauseMs(300);

printf("Connection established\n");

serFputs("This string is sent in data mode\r");

// Verify all data has been transmitted and switch to Command Mode
while(serFwrFree() != FOUTBUFSIZE);
BitWrPortI(PDDR, &PDDRShadow, 0, 5);
a7_serFwaitForString("\r>", 0);

printf("In Command Mode\n");

// Get local eb506 Bluetooth Address
serFputs("get address\r");
a7_serFwaitForString("ACK\r", 0);

// Read local address from get command
while(serFpeek() == -1) {}
serFread(szPortFData, 17, 100);
szPortFData[17] = '\r';
printf("Local eb506 address: %s\n", szPortFData);
a7_serFwaitForString("\r>", 0);

// Return to data mode
BitWrPortI(PDDR, &PDDRShadow, 1, 5);
a7_pauseMs(300);
```

```
// Send data through eb506
serFputs("My eb506 address is ");
serFputs(szPortFData);

// Verify all data has been transmitted and switch to Command Mode
while(serFwrFree() != FOUTBUFSIZE);
BitWrPortI(PDDR, &PDDRShadow, 0, 5);
a7_serFwaitForString("\r>", 0);

printf("In Command Mode\n");

// Disconnect from the remote device
serFputs("dis\r");
a7_serFwaitForString("\r>", 0);

printf("Disconnected\n");

serFclose();
```

Module Antenna Options

The eb506 module is available in two models. The eb506-AHC-IN features a high quality surface mount antenna and is designed for ease of integration when an external antenna is not required. For applications where an external antenna is required, such as those mounted in RF limiting enclosures or locations, the eb506-AHC-EN features an SMA coax connector allowing for cabled antenna solutions. This model provides a great deal of flexibility in antenna selection.

Hardware Connections

The eb506 module is designed to interface with a 3.3V CMOS signal environment. It supports a power supply of 3.3V and can be connected directly to RabbitCore modules RCM3000, RCM3100, RCM3200, and RCM3300. When inserting the eb506 module to any of the supported RabbitCore boards, it is important that Connectors CN1 and CN2 of the eb506 module are mated with the appropriate RabbitCore connectors. A full device pin out is available in the Technical Specifications section of this manual. A simple method of ensuring a proper connection is to line up the mounting holes of each board prior to insertion.

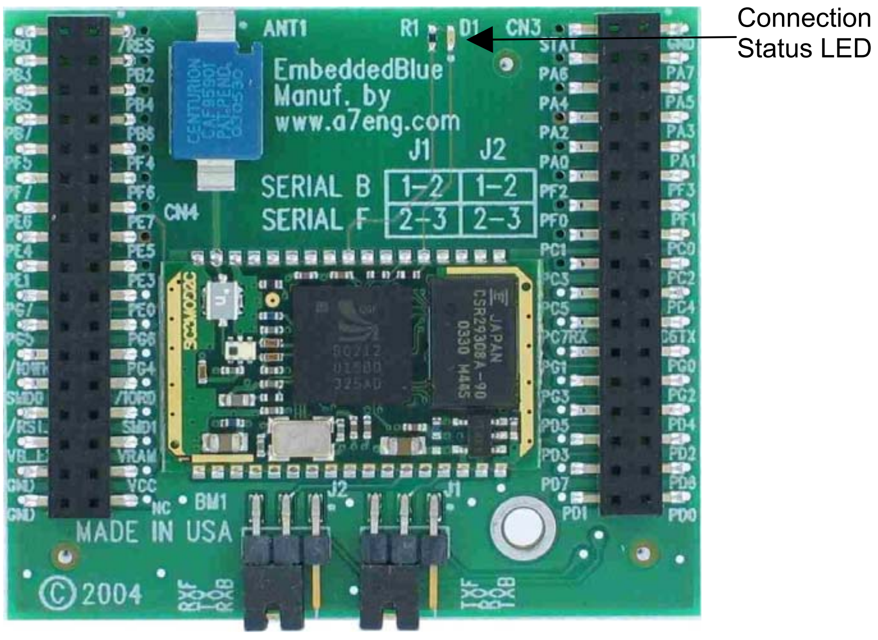


Figure 1: eb506 Module

BL2500

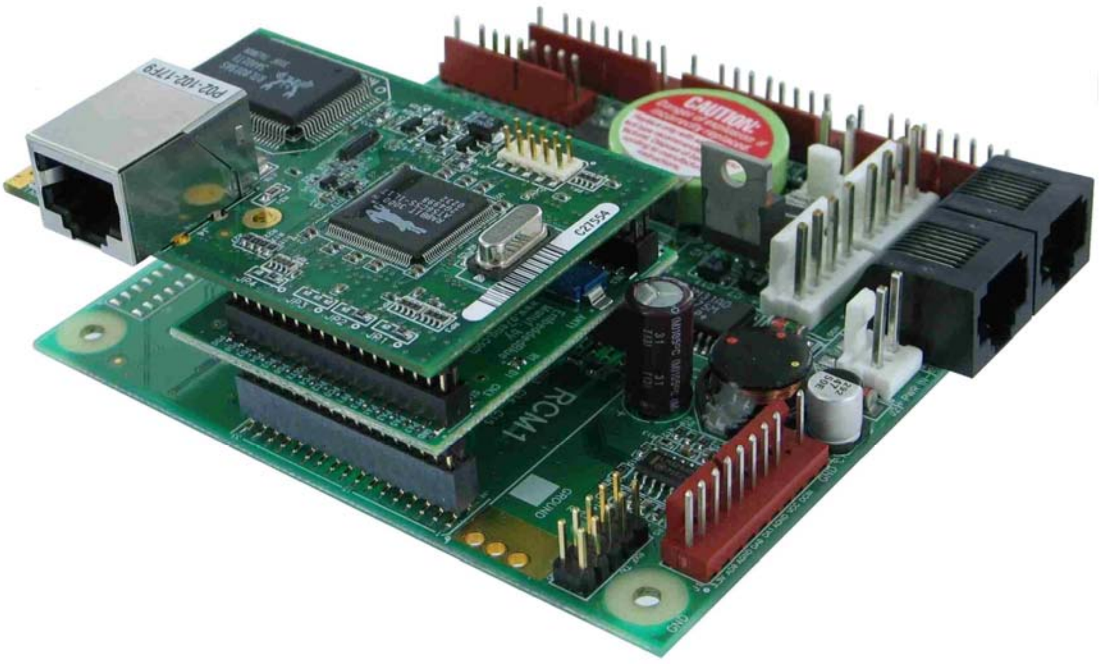


Figure 2: Z-World BL2500 Coyote SBC

The orientation of the installation is critical to prevent damage to the boards. Make sure that the mounting hole in the eb506 is lined up with the mounting hole in the RCM Module and that the installation is as shown in the figure above.

The Coyote BL2500 Single-Board Computer is compatible with the eb506 Bluetooth module. It can communicate with the RabbitCore module on either serial port B or F as configured by the jumpers J1 and J2. Install the eb506 adapter as pictured above, ensuring that both the eb506 module and the RabbitCore module are in the proper orientation by using the mounting holes as a guide.

RCM3000, RCM3010

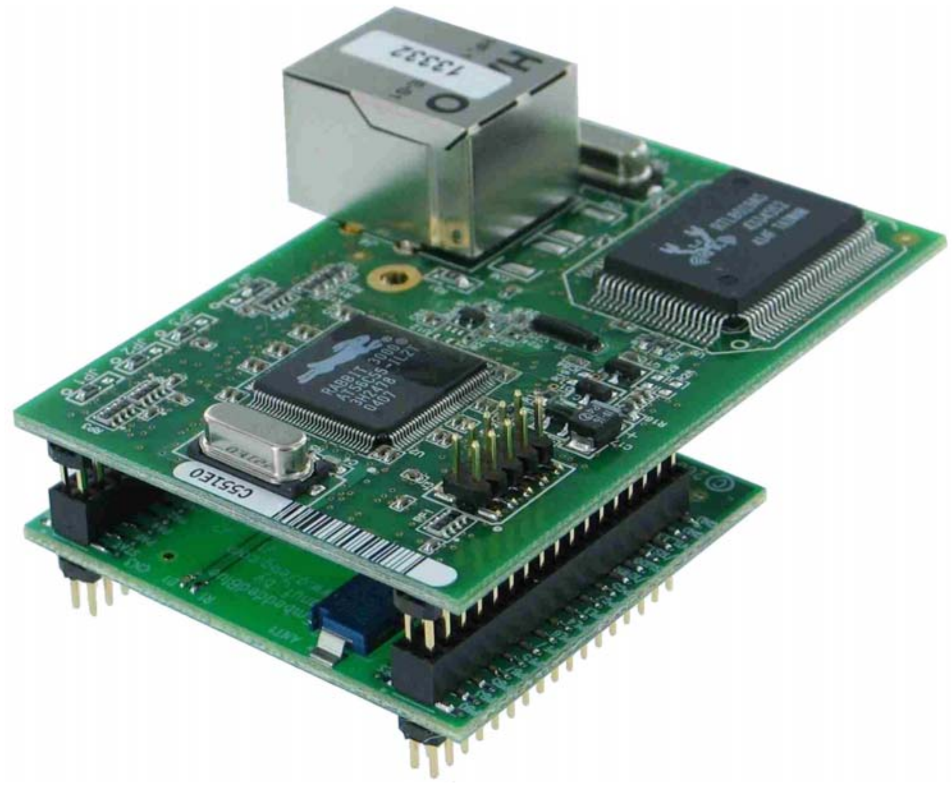


Figure 3: Z-World RCM3000 Module

⚠ **The orientation of the installation is critical to prevent damage to the boards. Make sure that the mounting hole in the eb506 is lined up with the mounting hole in the RCM Module and that the installation is as shown in the figure above.**

The RCM3000 and RCM3010 are compatible with the eb506 Bluetooth module. It can communicate with the RabbitCore module on either serial port B or F as configured by the jumpers J1 and J2. Install the eb506 adapter as pictured above, ensuring that both the eb506 module and the RabbitCore module are in the proper orientation by using the mounting holes as a guide.

RCM3100, RCM3110

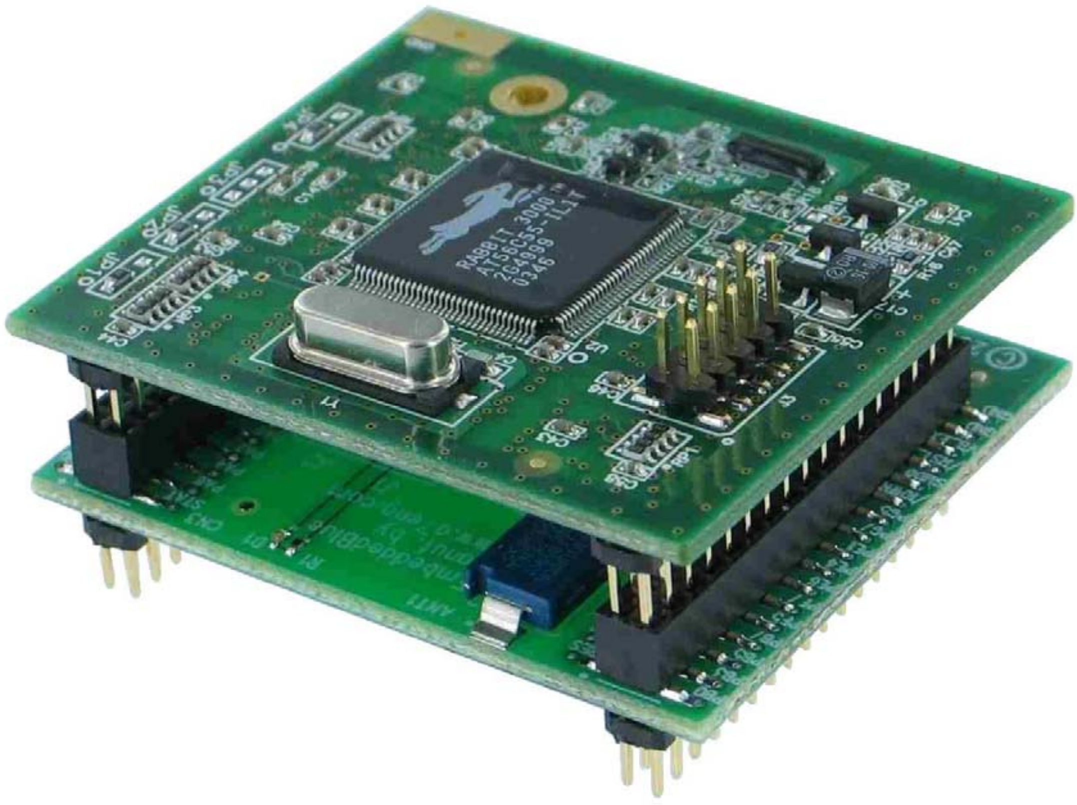


Figure 4: Z-World RCM3110 Module

The orientation of the installation is critical to prevent damage to the boards. Make sure that the mounting hole in the eb506 is lined up with the mounting hole in the RCM Module and that the installation is as shown in the figure above.

The RCM3100 and RCM3110 are compatible with the eb506 Bluetooth module. It can communicate with the RabbitCore module on either serial port B or F as configured by the jumpers J1 and J2. Install the eb506 adapter as pictured above, ensuring that both the eb506 module and the RabbitCore module are installed in the proper orientation by using the mounting holes as a guide.

RCM3200, RCM3210, RCM3220

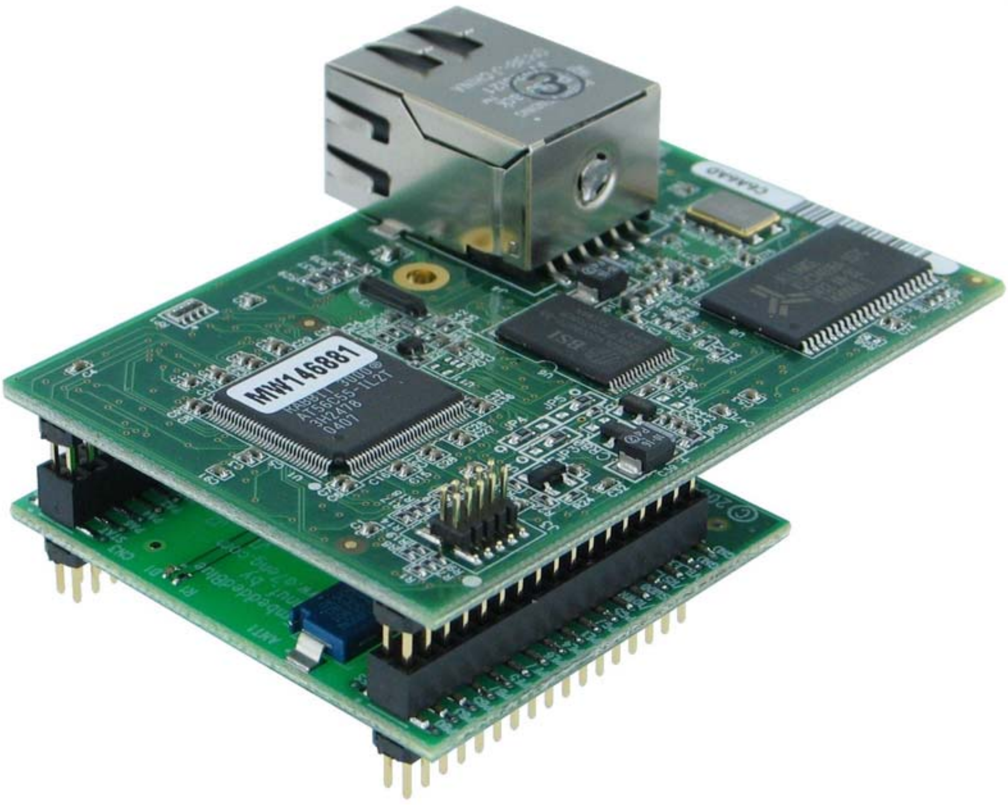


Figure 5: Z-World RCM3200 Module

⚠ **The orientation of the installation is critical to prevent damage to the boards. Make sure that the mounting hole in the eb506 is lined up with the mounting hole in the RCM Module and that the installation is as shown in the figure above.**

The RCM3200, RCM3210, and RCM3220 are compatible with the eb506 Bluetooth module. It can communicate with the RabbitCore module on either serial port B or F as configured by the jumpers J1 and J2. Install the eb506 adapter as pictured above, ensuring that both the eb506 module and the RabbitCore module are in the proper orientation by using the mounting holes as a guide.

RCM3300, RCM3310

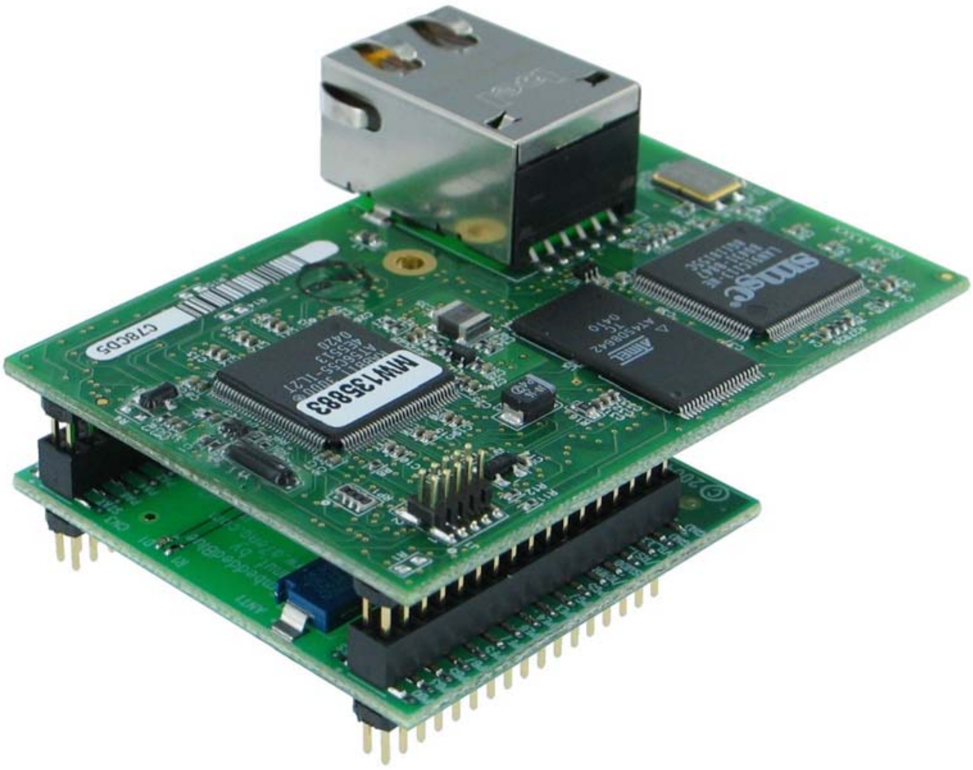


Figure 6: Z-World RCM3300 Module

The orientation of the installation is critical to prevent damage to the boards. Make sure that the mounting hole in the eb506 is lined up with the mounting hole in the RCM Module and that the installation is as shown in the figure above.

The RCM3300 and RCM3310 are compatible with the eb506 Bluetooth module. They must communicate with the eb506 on serial port F however, because port B is already in use by the on board serial flash. Use the jumpers J1 and J2 to configure the eb506 module to use port F. Install the eb506 adapter as pictured above, ensuring that both the eb506 module and the RabbitCore module are in the proper orientation by using the mounting holes as a guide.

If you are using the RCM3300 development board, there is an additional conflict between the eb506 module and the RS232 level translation chip U9. Both of these devices are configured to use serial port F and conflict on the use of PG3. To resolve this conflict, it is necessary to cut pin 26 (PG3) from CN1 on the bottom side of the eb506 module. This disconnects the serial port F receive line between the RCM3300 development board and the eb506 module. This pin should be cut and removed from connector CN1 to resolve the conflict. Please ensure that you have identified the proper pin before removal as shown in the picture below.

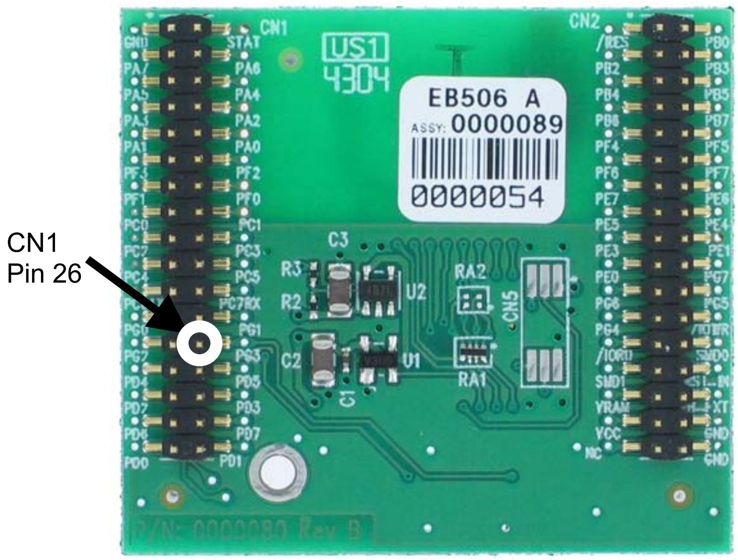


Figure 7: CN1 Pin 26 (PG3) of the eb506

RCM3360, RCM3370

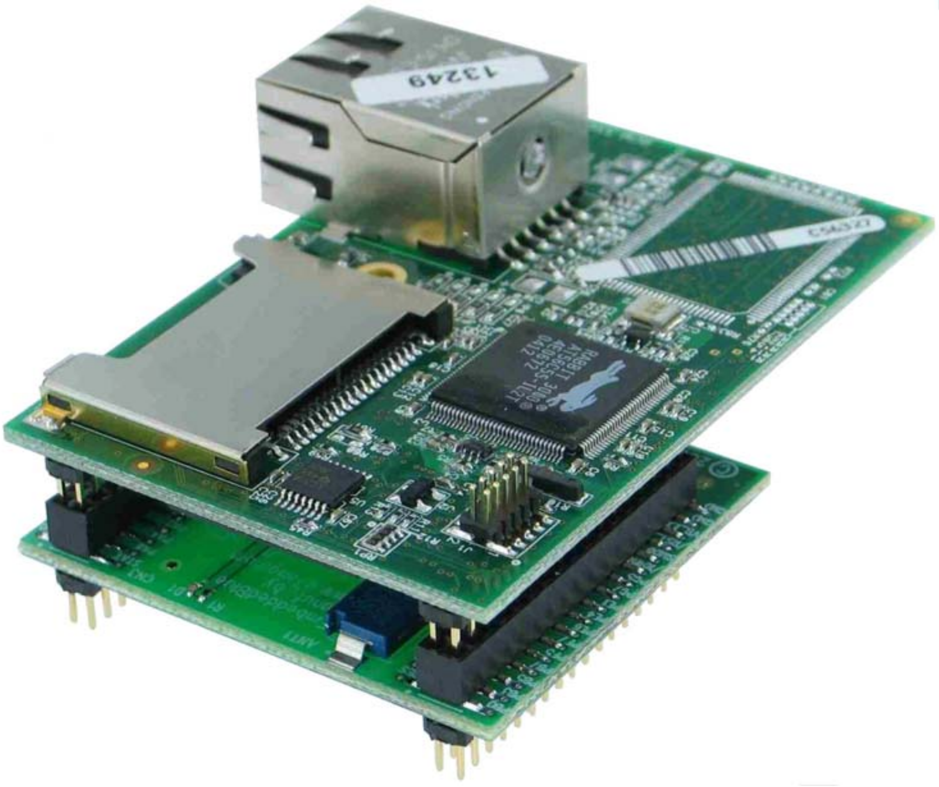


Figure 8: Z-World RCM3360 Module

The orientation of the installation is critical to prevent damage to the boards. Make sure that the mounting hole in the eb506 is lined up with the mounting hole in the RCM Module and that the installation is as shown in the figure above.

The RCM3360 and RCM3370 are compatible with the eb506 Bluetooth module. It can communicate with the RabbitCore module on either serial port B or F as configured by the jumpers J1 and J2. Install the eb506 adapter as pictured above, ensuring that both the eb506 module and the RabbitCore module are in the proper orientation by using the mounting holes as a guide.

Establishing a Connection

This section contains a number of exercises that demonstrate methods of establishing Bluetooth wireless connections with the eb506. The scenarios described are not meant to form an exhaustive list, but rather illustrate a number of more common and useful configurations. All source code shown in these exercises is available in electronic form on the accompanying CD, in the Samples folder, using the filename used in this manual. Additional samples will be made available on the A7 Engineering website at http://www.a7eng.com.

Connecting two eb506 Modules

In this exercise we will step through the process of establishing a connection between two eb506 boards plugged into separate RabbitCore Prototyping boards.

To perform this exercise, as documented, you will need two of the supported RabbitCore modules, two Prototyping boards, and two eb506 modules.

Step 1: Insert the eb506 Modules into the RCM and Prototyping Boards

In this step we will connect the eb506 modules, the prototyping boards, and the compatible RabbitCore modules together.

1. Insert an **eb506** module into the core module **MASTER SLOT** of the prototyping board assuring the module is inserted in the proper direction using the mounting hole as a guide.
2. Insert a **RabbitCore** module into the **eb506** module; assuring the module is inserted in the proper direction using the mounting hole as a guide. Repeat this step for the second set of boards.

Step 2: Use Pass-Through Prototyping to Get the eb506 Address

In this step we will configure one of the RabbitCore modules to use pass-through prototyping and retrieve its Bluetooth Device Address. This device will be referred to as device #2 for the rest of this sample.

1. Follow the **Pass-Through Prototyping** sample in the **eb506 basics** section of this manual.

   You should now be able to type commands and receive results directly from within a HyperTerminal window.

2. Using HyperTerminal, get the **address** of the **eb506** module by using the GET serial command.

   Example:

   ```
   >get address
   ACK
   00:0C:84:00:05:29
   >
   ```

Step 3: Connect the eb506 boards via Bluetooth

In this step we will develop and run a Dynamic C application on board #1 to establish a connection with board #2.

1. Using the Dynamic C Editor; on the **File** menu, click **New**.

   This will create a new project window within the Dynamic C Editor.

2. Enter the following **program code** into the editor, replacing the Bluetooth device address with the device address of the eb506 on board #2, which we obtained in the previous step. This application is available in electronic form on the accompanying CD, in the Samples folder, in the file Connect.c.

```
serFopen(9600);
a7_serFeb506Init();

// Connect to the remote device
serFputs("con 00:0C:84:00:05:29\r");
a7_serFwaitForString("ACK\r", 0);

// Wait for the connection to be established and switch to data mode
while(!BitRdPortI(PDDR, 4));
BitWrPortI(PDDR, &PDDRShadow, 1, 5);
a7_pauseMs(300);

// Wait for twenty seconds
```

```
a7_pauseMs(20000);

// Verify all data has been transmitted and switch to Command Mode
while(serFwrFree() != FOUTBUFSIZE);
BitWrPortI(PDDR, &PDDRShadow, 0, 5);
a7_serFwaitForString("\r>", 0);

// Disconnect from the remote device
serFputs("dis\r");
a7_serFwaitForString("\r>", 0);
```

The Dynamic C application establishes a connection with the remote Bluetooth device, waits twenty seconds, switches back to command mode and then disconnects from the remote device.

3. On the **File** menu, click **Save As**.
4. In the **File name** box, enter a file name to which to save the program just created. For example, Connect.c.
5. Click **Save**.
6. Apply **power** to board #1 and board #2.
7. On the **Compile** menu, click **Compile to Target->Compile to Flash**
8. When the compile is complete, click the **Run** icon.

   The Connection Status LED (see Figure 1 on page 19) on both eb506 modules will turn on when a connection is established between the two eb506 modules.
9. Disconnect the **power** from board #1.
10. Disconnect the **power** from board #2.

Connecting a PC with a DBT-120 to an RCM3110

In this exercise we will step through the process of establishing a connection from a PC that has a D-Link® DBT-120 Bluetooth USB Adapter to an eb506 module in pass through mode.

To perform this exercise, as documented, you will need a D-Link DBT-120, a Z-World RCM3110 module, a Z-World RabbitCore prototyping board, and an eb506 module. If you are using any of the other supported Z-World boards, you may need to make adjustments to this exercise.

On the PC, the DBT-120 Bluetooth Software associates a COM port for establishing a connection from the PC to a remote Bluetooth device and a separate COM port for connections that are established from a remote Bluetooth device to the PC. This exercise demonstrates establishing a connection from the PC to a remote eb506. The next exercise will demonstrate establishing a connection from a remote eb506 to the PC.

The D-Link DBT-120 Bluetooth USB Adapter software must be fully installed prior to establishing a connection. The PC settings shown in this exercise are based upon the software provided with the D-Link DBT-120 Bluetooth USB Adapter.

Step 1: DBT-120 Setup

In this step we will attach the DBT-120 USB Adapter to the PC. The software for the DBT-120 should already be setup.

1. Connect the **DBT-120** to an available **USB port** on the PC, following the instructions provided with the DBT-120 Bluetooth USB Adapter.

Step 2: Z-World Prototyping Board – eb506 Setup

In this step we will connect the eb506 module, the prototyping board, and the RCM3110 module together.

1. Insert an **eb506** module into the core module **MASTER SLOT** of the prototyping board assuring the module is inserted in the proper direction using the mounting hole as a guide.
2. Insert the **RCM3110** module into the **eb506** module; assuring that the module is inserted in the proper direction using the mounting hole as a guide.
3. Apply **power** to the **Prototyping Board**.

   Power can be applied by attaching the AC-Adapter provided by Z-World.

Step 3: Establishing trust between the PC and the eb506 module

In this step we will establish a trusted relationship between the PC and the eb506 module. Verifying passkeys is required when performing the initial connection on two devices that require security. If you have connected these two devices once with security enabled then this step should be skipped.

The actions in this step need to be performed only once for the eb506. After performing the actions in this step, the connection security details will be stored on both the PC and the eb506 module. Therefore, future connections can be established to an eb506 by simply opening the associated COM port.

1. Open the **Bluetooth Devices** dialog by double-clicking on the Bluetooth tray icon.

   This will display the Bluetooth Devices dialog.

2. Click **Find Bluetooth Devices** to locate the **eb506 module** connected to the prototyping board.

   Provided the eb506 on the prototyping board is within range, eb506 will be shown in the window.

3. Right click on the eb506 and click **Connect A7 Serial Port**.

   This will bring up the Bluetooth PIN Code Request dialog since we have not connected to this device in the past.

4. Enter the Bluetooth **PIN Code** for the eb506.

   The factory default passkey is **0000**. The eb506 module enables security by default so the devices must verify passkeys to establish a trusted relationship before they can communicate.

   Note about terminology: The DBT-120 software uses the term “PIN Code” as a substitute to the eb506’s “Passkey”. These two terms refer to the same idea of using a secret code to establish a connection.

Step 4: Establish a Connection Using the DBT-120 Bluetooth Software

In this step we will establish a connection from the PC to the eb506 module inserted into the prototyping board.

The actions in this step need to be performed only once for the eb506. After performing the actions in this step, the connection details will be stored on the PC. Therefore, future connections can be established to an eb506 by simply opening the associated COM port.

1. Open **My Bluetooth Places** by double-clicking on the desktop icon.

2. Click **Find Bluetooth Devices** to locate the **eb506 module** connected to the prototyping board.

   Provided the eb506 on the prototyping board is within range, eb506 will be shown in the window.

3. Right click on **eb506** and click **Discover Available services**.

   The A7 Serial Port service will be shown in the window.

4. Right click on **A7 Serial Port** and click **Connect to Bluetooth Serial Port**.

   This will establish a connection from the PC to the eb506 on the prototyping board and associate this connection with a specific COM port.

5. If the **A7 Serial Port** dialog is shown, click **OK**.

6. Right click on **A7 Serial Port on eb506** and click **Properties**.

   This will display the Bluetooth Properties dialog.

7. In the **Port** dropdown, which is disabled, please note the **COM port** shown.

   The DBT-120 Bluetooth software associates a specific COM port for a connection from the PC to an eb506. Applications, such as HyperTerminal, use this COM port to establish a connection and communicate with an eb506 from the PC. Remember, this COM port is used to establish a connection from the PC to the eb506. A different COM port is used when a connection is established from the eb506 to the PC.

8. Click **OK**.

9. Select **A7 Serial Port** and click **Disconnect Bluetooth Serial Port**.

   This will disconnect the wireless connection to the eb506 on the prototyping board.

Step 5: HyperTerminal Setup

In this step we will setup the Windows HyperTerminal application to establish a connection with the eb506 on the prototyping board.

1. Open **HyperTerminal**.

   This will display the Connection Description dialog.

2. In the **Name** box, type the name of your connection. For example, eb506-Proto.

3. Click **OK**.

   This will display the Connect To dialog.

4. In the **Connect using** dropdown, select the **serial port** associated with the DBT-120 Bluetooth connection discovered in the previous step.
5. Click **OK**.

   This will display the Properties dialog.
6. In the **Bits per second** dropdown, select **9600**.
7. In the **Data bits** dropdown, select **8**.
8. In the **Parity** dropdown, select **None**.
9. In the **Stop bits** dropdown, select **1**.
10. In the **Flow control** dropdown, select **None**.
11. Click **OK**.

    This will establish a connection with the eb506 on the prototyping board.
12. On the **Call** menu, click **Disconnect**.

    This will disconnect the connection just established, so that we can modify the connection properties in the following actions.
13. On the **File** menu, click **Properties**.

    This will display the Properties dialog.
14. On the **Settings** tab, click **ASCII Setup**.

    This will display the ASCII Setup dialog.
15. Check the **Send line ends with line feeds** checkbox.
16. Check the **Echo typed characters locally** checkbox.
17. Check the **Append line feeds to incoming line ends** checkbox.
18. Check the **Wrap lines that exceed terminal width** checkbox.
19. Click **OK**.

    This will return to the Properties dialog.
20. Click **OK**.

Step 6: Establish a Connection Using HyperTerminal

In this step we will establish a connection from the PC to the eb506 on the prototyping board using HyperTerminal. This step relies on the connection information created previously.

1. On the **Call** menu, click **Call**.

   This will establish a connection with the eb506 on the prototyping board. The Connection Status LED (see Figure 1 on page 19) on the eb506 module will turn on when a connection is established.

2. On the **Call** menu, click **Disconnect**.

   This will close the connection with the eb506 on the prototyping board.

Connecting an RCM3110 to a PC with a DBT-120

In this exercise we will step through the process of establishing a connection from an eb506 module inserted into a Z-World prototyping board to a PC that has a D-Link® DBT-120 Bluetooth USB Adapter.

To perform this exercise, as documented, you will need a D-Link DBT-120, a Z-World RCM3110 module, a Z-World RabbitCore prototyping board, and an eb506 module. If you are using any of the other supported Z-World boards, you may need to make adjustments to this exercise.

On the PC, the DBT-120 Bluetooth Software associates a COM port for establishing a connection from the PC to a remote Bluetooth device and a separate COM port for connections that are established from a remote Bluetooth device to the PC. This exercise demonstrates establishing a connection from a remote eb506 to the PC. When a remote Bluetooth device establishes a connection with the PC, the connection is established with the DBT-120 Bluetooth USB Adapter software running on the PC. To gain access to the data, an application, such as HyperTerminal, must open the COM port associated with the connection established from the remote device. In the Communications section, we will step through this process.

The D-Link DBT-120 Bluetooth USB Adapter software must be fully installed prior to establishing a connection. The PC settings shown in this exercise are based upon the software provided with the D-Link DBT-120 Bluetooth USB Adapter.

Step 1: DBT-120 Setup

In this step we will attach the DBT-120 USB Adapter to the PC. The software for the DBT-120 should already be setup.

1. Connect the **DBT-120** to an available **USB port** on the PC, following the instructions provided with the DBT-120 Bluetooth USB Adapter.

Step 2: Obtain the Bluetooth Address of the PC

In this step we will obtain the Bluetooth address of the DBT-120 USB Adapter attached to the PC.

1. Open **My Bluetooth Places** by double-clicking on the desktop icon.
2. Right click on **My Device** and click **Properties**.

   This will display the Bluetooth Configuration dialog.

3. Select the **Hardware** tab and note the **Device Address** shown in the Device Properties section of the dialog.

   The device address will be used in the Dynamic C application developed in the next step.

4. Click **Cancel**

   This will close the Bluetooth Configuration dialog.

Step 3: Write a Dynamic C Application to Connect to the PC

In this step we will attach an eb506 module to the prototyping board and develop a Dynamic C application to establish a connection with the PC.

1. Insert an **eb506** module into the core module **MASTER SLOT** of the prototyping board assuring the module is inserted in the proper direction using the mounting hole as a guide.
2. Insert the **RCM3110** module into the **eb506** module; assuring that the module is inserted in the proper direction using the mounting hole as a guide.
3. Connect the **RCM3110 programming port** to the **PC**.
4. Open the **Dynamic C Editor**.
5. Enter the following **program code** into the editor, replacing the Bluetooth device address with the device address of the PC, which we obtained from the Hardware tab of the Device Properties section of the Bluetooth Configuration dialog in the previous step. This application is available in electronic form on the accompanying CD, in the Samples folder, in the file Connect.c.

```
serFopen(9600);
a7_serFeb506Init();

// Connect to the PC
serFputs("con 00:0C:84:00:05:30\r");
a7_serFwaitForString("ACK\r", 0);

// Wait for the connection to be established and switch to data mode
while(!BitRdPortI(PDDR, 4));
BitWrPortI(PDDR, &PDDRShadow, 1, 5);
a7_pauseMs(300);
```

```
// Wait for twenty seconds
a7_pauseMs(20000);

// Verify all data has been transmitted and switch to Command Mode
while(serFwrFree() != FOUTBUFSIZE);
BitWrPortI(PDDR, &PDDRShadow, 0, 5);
a7_serFwaitForString("\r>", 0);

// Disconnect from the remote device
serFputs("dis\r");
a7_serFwaitForString("\r>", 0);
```

The Dynamic C application establishes a connection with the PC device, waits twenty seconds, switches back to command mode and then disconnects from the PC.

6. On the **File** menu, click **Save As**.
7. In the **File name** box, enter a file name to which to save the program just created. For example, Connect.c.
8. Click **Save**.

Step 4: Establishing trust between the PC and the eb506 module

In this step we will establish a trusted relationship between the PC and the eb506 module. Verifying passkeys is required when performing the initial connection on two devices that require security. If you have connected these two devices once with security enabled then this step should be skipped.

The actions in this step need to be performed only once for the eb506. After performing the actions in this step, the connection security details will be stored on both the PC and the eb506 module. Therefore, future connections can be established to an eb506 by simply opening the associated COM port.

9. Open the **Bluetooth Devices** dialog by double-clicking on the Bluetooth tray icon.

   This will display the Bluetooth Devices dialog.
10. Click **Find Bluetooth Devices** to locate the **eb506 module** connected to the prototyping board.

    Provided the eb506 on the prototyping board is within range, eb506 will be shown in the window.
11. Right click on the eb506 and click **Connect A7 Serial Port**.

This will bring up the Bluetooth PIN Code Request dialog since we have not connected to this device in the past.

12. Enter the Bluetooth **PIN Code** for the eb506.

    The factory default passkey is **0000**. The eb506 module enables security by default so the devices must verify passkeys to establish a trusted relationship before they can communicate.

    Note about terminology: The DBT-120 software uses the term “PIN Code” as a substitute to the eb506’s “Passkey”. These two terms refer to the same idea of using a secret code to establish a connection.

Step 5: Connect the eb506 on the Prototyping Board to the PC

1. Apply **power** to the **prototyping board**.

   Power can be applied by attaching the AC-Adapter provided by Z-World.

2. On the **Run** menu, click **Run**.

   The Connection Status LED (see Figure 1 on page 19) on the eb506 module will turn on when a connection is established. Additionally, on the My Bluetooth Places window, in the Additional Information column, the text “Connected” will be shown while a connection exists between the eb506 and the PC.

Connecting a PC with XP SP2 to an RCM3110

In this exercise we will step through the process of establishing a connection from a PC with a Bluetooth USB adapter that is running Windows XP SP2 to an eb506 module in pass-through mode.

To perform this exercise, as documented, you will need a PC running Windows XP SP2, a Bluetooth USB adapter, a Z-World RCM3110 module, a Z-World RabbitCore prototyping board, and an eb506 module. If you are using any of the other supported Z-World boards, you may need to make adjustments to this exercise.

On the PC, the Microsoft Bluetooth Software associates a COM port for establishing a connection from the PC to a remote Bluetooth device and a separate COM port for connections that are established from a remote Bluetooth device to the PC. This exercise demonstrates establishing a connection from the PC to a remote eb506. The next exercise will demonstrate establishing a connection from a remote eb506 to the PC.

Step 1: Bluetooth USB Adapter Setup

In this step we will attach the Bluetooth USB Adapter to the PC. Windows XP SP2 should automatically detect and configure the adapter for use.

1. Connect the **Bluetooth USB Adapter** to an available **USB port** on the PC.

Step 2: Z-World Prototyping Board – eb506 Setup

In this step we will connect the eb506 module, the prototyping board, and the RCM3110 module together.

1. Insert an **eb506** module into the core module **MASTER SLOT** of the prototyping board assuring the module is inserted in the proper direction using the mounting hole as a guide.
2. Insert the **RCM3110** module into the **eb506** module; assuring that the module is inserted in the proper direction using the mounting hole as a guide.
3. Apply **power** to the **Prototyping Board**.

   Power can be applied by attaching the AC-Adapter provided by Z-World.

Step 3: Establishing trust between the PC and the eb506 module

In this step we will establish a trusted relationship between the PC and the eb506 module. Verifying passkeys is required when performing the initial connection on two devices that require security. If you have connected these two devices once with security enabled then this step should be skipped.

The actions in this step need to be performed only once for the eb506. After performing the actions in this step, the connection security details will be stored on both the PC and the eb506 module. Therefore, future connections can be established to an eb506 by simply opening the associated COM port.

1. Open the **Bluetooth Devices** dialog by double-clicking on the Bluetooth tray icon.

   This will display the Bluetooth Devices dialog.

2. Click **Add** to open the Add Bluetooth Device Wizard

   The Windows XP SP2 Bluetooth Software requires that devices are added before they can be used.

3. Click **My device is set up and ready to be found** and then click **Next** to locate the eb506 module connected to the prototyping board.

   Provided the eb506 on the prototyping board is within range, eb506 will be shown in the window.

4. Select **eb506** and click **Next**.

   The passkey selection dialog will be shown. The eb506 module enables security by default so the devices must verify passkeys to establish a trusted relationship before they can communicate.

5. Select **Use the passkey found in the documentation,** enter **0000** into the edit field, and click **Next**.

   This will establish a connection from the PC to the eb506 on the prototyping board and associate this connection with a specific COM port.

6. Please note both the Outgoing and Incoming **COM ports** shown.

   The Microsoft Bluetooth software associates a specific COM port for a connection from the PC to an eb506. Applications, such as HyperTerminal, use this COM port to establish a connection and communicate with an eb506 from the PC. Remember, this COM port is used to establish a connection from the PC to the eb506. A different COM port is used when a connection is established from the eb506 to the PC.

7. Click **Finish**.

   This will complete the wizard and close the Add Bluetooth Device Wizard.

Step 4: HyperTerminal Setup

In this step we will setup the Windows HyperTerminal application to establish a connection with the eb506 on the prototyping board.

1. Open **HyperTerminal**.

   This will display the Connection Description dialog.

2. In the **Name** box, type the name of your connection. For example, eb506-Proto.
3. Click **OK**.

   This will display the Connect To dialog.

4. In the **Connect using** dropdown, select the **serial port** to which the Microsoft Bluetooth software associated with the connection from the PC to the eb506 on the prototype board.

   The COM port associated with the connection was discovered in the previous step.

5. Click **OK**.

   This will display the Properties dialog.

6. In the **Bits per second** dropdown, select **9600**.
7. In the **Data bits** dropdown, select **8**.
8. In the **Parity** dropdown, select **None**.
9. In the **Stop bits** dropdown, select **1**.
10. In the **Flow control** dropdown, select **None**.
11. Click **OK**.

    This will establish a connection with the eb506 on the prototyping board.

12. On the **Call** menu, click **Disconnect**.

    This will disconnect the connection just established, so that we can modify the connection properties in the following actions.

13. On the **File** menu, click **Properties**.

    This will display the Properties dialog.

14. On the **Settings** tab, click **ASCII Setup**.

    This will display the ASCII Setup dialog.

15. Check the **Send line ends with line feeds** checkbox.

16. Check the **Echo typed characters locally** checkbox.
17. Check the **Append line feeds to incoming line ends** checkbox.
18. Check the **Wrap lines that exceed terminal width** checkbox.
19. Click **OK**.

    This will return to the Properties dialog.
20. Click **OK**.

Step 5: Establish a Connection from the PC Using HyperTerminal

In this step we will establish a connection from the PC to the eb506 on the prototyping board, using HyperTerminal. This step relies on the connection information created previously in Step 3.

1. On the **Call** menu, click **Call**.

   This will establish a connection with the eb506 on the prototyping board. The Connection Status LED (see Figure 1 on page 19) on the eb506 module will turn on when a connection is established. By default, the eb506 is in data mode.
2. On the **Call** menu, click **Disconnect**.

   This will close the connection with the eb506 on the prototyping board.

Connecting an RCM3110 to a PC with XP SP2

In this exercise we will step through the process of establishing a connection from an eb506 module inserted into a Z-World prototyping board to a PC that is running Windows XP SP2 and has a Bluetooth USB Adapter.

To perform this exercise, as documented, you will need a PC running Windows XP SP2, a Bluetooth USB Adapter, a Z-World RCM3110 module, a Z-World RabbitCore prototyping board, and an eb506 module. If you are using any of the other supported Z-World boards, you may need to make adjustments to this exercise.

On the PC, the Microsoft Bluetooth Software associates a COM port for establishing a connection from the PC to a remote Bluetooth device and a separate COM port for connections that are established from a remote Bluetooth device to the PC. This exercise demonstrates establishing a connection from a remote eb506 to the PC. When a remote Bluetooth device establishes a connection with the PC, the connection is established with the Bluetooth USB Adapter and the software running on the PC. To gain access to the data, an application, such as HyperTerminal, must open the COM port associated with the connection established from the remote device. In the Communications section, we will step through this process.

Step 1: Bluetooth USB Adapter Setup

In this step we will attach the Bluetooth USB Adapter to the PC. Windows XP SP2 should automatically detect and configure the adapter for use.

1. Connect the **Bluetooth USB Adapter** to an available **USB port** on the PC.

Step 2: Obtain the Bluetooth Address of the PC

In this step we will obtain the Bluetooth address of the Bluetooth USB Adapter attached to the PC.

1. Open the **Bluetooth Devices** dialog by double-clicking on the Bluetooth tray icon.

   This will display the Bluetooth Devices dialog.

2. On the **Hardware** tab select **Generic Bluetooth Radio** and click on the **Properties** button.

   This will display the Generic Bluetooth Radio Properties dialog.

3. Select the **Advanced** tab and note the **Address** shown in the Radio Information section of the dialog.

   The device address will be used in the Dynamic C application developed in the next step.

4. Click **OK**

   This will close the Generic Bluetooth Radio Properties dialog.

5. Click **OK**

   This will close the Bluetooth Devices dialog.

Step 3: Establishing trust between the PC and the eb506 module

In this step we will establish a trusted relationship between the PC and the eb506 module. Verifying passkeys is required when performing the initial connection on two devices that require security. If you have already connected these two devices with security enabled then this step should be skipped.

The actions in this step need to be performed only once for the eb506. After performing the actions in this step, the connection security details will be stored on both the PC and the eb506 module. Therefore, future connections can be established to an eb506 by simply opening the associated COM port.

1. Open the **Bluetooth Devices** dialog by double-clicking on the Bluetooth tray icon.

   This will display the Bluetooth Devices dialog.

2. Click **Add** to open the Add Bluetooth Device Wizard

   The Windows XP SP2 Bluetooth Software requires that devices are added before they can be used.

3. Click **My device is set up and ready to be found** and then click **Next** to locate the eb506 module connected to the prototyping board.

   Provided the eb506 on the prototyping board is within range, eb506 will be shown in the window.

4. Select **eb506** and click **Next**.

   The passkey selection dialog will be shown. The eb506 module enables security by default so the devices must verify passkeys to establish a trusted relationship before they can communicate.

5. Select **Use the passkey found in the documentation,** enter **0000** into the edit field, and click **Next**.

   This will establish a connection from the PC to the eb506 on the prototyping board and associate this connection with a specific COM port.

6. Please note both the Outgoing and Incoming **COM ports** shown.

The Microsoft Bluetooth software associates a specific COM port for a connection from the PC to an eb506. Applications, such as HyperTerminal, use this COM port to establish a connection and communicate with an eb506 from the PC. Remember, this COM port is used to establish a connection from the PC to the eb506. A different COM port is used when a connection is established from the eb506 to the PC.

7. Click **Finish**.

   This will complete the wizard and close the Add Bluetooth Device Wizard.

Step 4: Write a Dynamic C Application to Connect to the PC

In this step we will attach an eb506 module to the prototyping board and develop a Dynamic C application to establish a connection with the PC.

1. Insert an **eb506** module into the core module **MASTER SLOT** of the prototyping board assuring the module is inserted in the proper direction using the mounting hole as a guide.
2. Insert the **RCM3110** module into the **eb506** module; assuring that the module is inserted in the proper direction using the mounting hole as a guide.
3. Connect the **RCM3110 programming port** to the **PC**.
4. Open the **Dynamic C Editor**.
5. Enter the following **program code** into the editor, replacing the Bluetooth device address with the device address of the PC, which we obtained from the Hardware tab of the Device Properties section of the Bluetooth Configuration dialog in the previous step. This application is available in electronic form on the accompanying CD, in the Samples folder, in the file Connect.c.

```
serFopen(9600);
a7_serFeb506Init();

// Connect to the PC
serFputs("con 00:0C:84:00:05:30\r");
a7_serFwaitForString("ACK\r", 0);

// Wait for the connection to be established and switch to data mode
while(!BitRdPortI(PDDR, 4));
BitWrPortI(PDDR, &PDDRShadow, 1, 5);
a7_pauseMs(300);
```

```
// Wait for twenty seconds
a7_pauseMs(20000);

// Verify all data has been transmitted and switch to Command Mode
while(serFwrFree() != FOUTBUFSIZE);
BitWrPortI(PDDR, &PDDRShadow, 0, 5);
a7_serFwaitForString("\r>", 0);

// Disconnect from the remote device
serFputs("dis\r");
a7_serFwaitForString("\r>", 0);
```

The Dynamic C application establishes a connection with the PC, waits twenty seconds, switches back to command mode and then disconnects from the PC.

6. On the **File** menu, click **Save As**.
7. In the **File name** box, enter a file name to which to save the program just created. For example, Connect.c.
8. Click **Save**.

Step 5: Connect the eb506 on the Prototyping Board to the PC

1. Apply **power** to the **prototyping board**.

   Power can be applied by attaching the AC-Adapter provided by Z-World.

2. On the **Run** menu, click **Run**.

   The Connection Status LED (see Figure 1 on page 19) on the eb506 module will turn on when a connection is established.

Connecting a Pocket PC 2003 device to an RCM3110

In this exercise we will step through the process of establishing a connection from a Pocket PC 2003 device with integrated Bluetooth, to an eb506 module inserted into a RabbitCore prototyping board.

To perform this exercise, as documented, you will need a Pocket PC 2003 device, an RCM3110 module, a prototyping board, and an eb506 module. Depending on the specific Pocket PC model that you are using, you may need to make minor adjustments to this exercise.

Step 1: Z-World Prototyping Board – eb506 Setup

In this step we will connect the eb506 module, the prototyping board, and the RCM3110 module together.

1. Insert an **eb506** module into the core module **MASTER SLOT** of the prototyping board assuring the module is inserted in the proper direction using the mounting hole as a guide.
2. Insert the **RCM3110** module into the **eb506** module; assuring that the module is inserted in the proper direction using the mounting hole as a guide.
3. Apply **power** to the **Prototyping Board**.

   Power can be applied by attaching the AC-Adapter provided by Z-World.

Step 2: Pocket PC 2003 Setup

In this step we will setup the Pocket PC for connecting to the eb506.

1. Tap the **Bluetooth icon** in the system tray on the Today screen and select **Bluetooth Manager**.

   This will display the Bluetooth Manager dialog.
2. On the **New** menu, select **Connect**.

   This will display the first page of the Connection Wizard.
3. Select **Explore a Bluetooth device** and tap **Next**.

   This will display the next page of the Connection Wizard.
4. Tap in the **Device** box.

   This will display the Connection Wizard Bluetooth Browser dialog containing a list of found devices.

5. Tap **eb506**.

   This will display the next page of the Connection Wizard.

6. In the **Service Selection** box, select **A7 Serial Port**.

7. Tap **Next**.

   This will create a shortcut for the service.

8. Tap **Finish**.

   This will display the Bluetooth Manager dialog with the shortcut created in the window, eb506: A7 Serial Port.

Step 3: Establish a Connection

In this step we will establish a connection from the Pocket PC to the eb506.

1. Tap-and-hold the shortcut created in the previous step, **eb506: A7 Serial Port**.

2. Select **Connect**.

   This will establish a connection with the eb506 on the RabbitCore Prototyping board. The Connection Status LED (see Figure 1 on page 19) on the eb506 module will turn on when a connection is established.

3. Tap **Active Connections**.

   This will display the Bluetooth Manager Active Connections page showing the status of your active Bluetooth connections.

4. Tap **My Shortcuts**.

5. Tap-and-hold the shortcut created in the previous step, **eb506: A7 Serial Port**.

6. Select **Disconnect**.

   This will close the connection with the eb506 on the RabbitCore Prototyping board. The Connection Status LED on the eb506 module will turn off.

Connecting an RCM3110 to a Pocket PC 2003 device

In this exercise we will step through the process of establishing a connection from an eb506 module attached to a RabbitCore prototyping board to a Pocket PC 2003 device with integrated Bluetooth.

To perform this exercise, as documented, you will need a Pocket PC 2003 device, an RCM3110 module, a RabbitCore prototyping board, and an eb506 module. Depending on the specific Pocket PC model that you are using, you may need to make minor adjustments to this exercise.

Step 1: Obtain the Bluetooth Address of the Pocket PC

In this step we will obtain the Bluetooth address of the Pocket PC.

1. Tap the **Bluetooth icon** in the system tray on the Today screen and select **Bluetooth Settings**.

   This will display the Settings dialog.

2. Tap the **Accessibility** tab and note the **Address** shown in the Device Identification section of the dialog.

   The device address will be used in the Dynamic C application developed in the next step.

3. Tap **OK** to close the dialog.

Step 2: Write a Dynamic C Application to Connect to the Pocket PC

In this step we will attach an eb506 module to the RabbitCore board and develop a Dynamic C application to establish a connection with the Pocket PC.

1. Insert an **eb506** module into the **Master Module Connectors** of the Prototyping board, assuring that CN1 of the eb506 module is inserted into connector JA of the Prototyping board.

2. Connect the **RabbitCore board serial port** to the **PC**.

3. Open the **Dynamic C Editor**.

4. Enter the following **program code** into the editor, replacing the Bluetooth device address with the device address of the Pocket PC, which we obtained in the previous step. This application is available in electronic form on the accompanying CD, in the Samples folder, in the file Connect.c.

```
serFopen(9600);

a7_serFeb506Init();
```

```
// Connect to the Pocket PC
serFputs("con 00:0C:84:00:05:31\r");
a7_serFwaitForString("ACK\r", 0);

// Wait for the connection to be established and switch to data mode
while(!BitRdPortI(PDDR, 4));
BitWrPortI(PDDR, &PDDRShadow, 1, 5);
a7_pauseMs(300);

// Wait for twenty seconds
a7_pauseMs(20000);

// If the connection is still valid
If(BitRdPortI(PDDR, 4))
{
   // Verify all data has been transmitted and switch to Command Mode
   while(serFwrFree() != FOUTBUFSIZE);
   BitWrPortI(PDDR, &PDDRShadow, 0, 5);
   a7_serFwaitForString("\r>", 0);

   // Disconnect from the remote device
   serFputs("dis\r");
   a7_serFwaitForString("\r>", 0);
}
```

The Dynamic C application establishes a connection with the remote Bluetooth device, waits twenty seconds, switches back to command mode and then disconnects from the remote device.

On most Pocket PC 2003 devices, the Bluetooth software closes the connection after a short period of time if there is not an application running on the device to receive the data over the connection. Therefore, after the twenty second wait, the Dynamic C application checks to see if there is still a valid connection before switching to Command Mode. If there is no connection, the eb506 is already in Command Mode.

5. On the **File** menu, click **Save As**.

6. In the **File name** box, enter a file name to which to save the program just created. For example, Connect.c.
7. Click **Save**.

Step 3: Establish a Connection

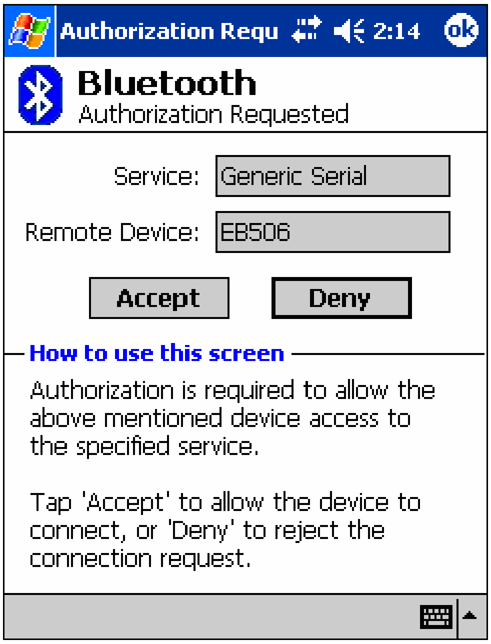
In this step we will establish a connection from the RabbitCore board to the Pocket PC.

1. Turn on the **Pocket PC 2003 device**.
2. Tap the **Bluetooth icon** and select **Bluetooth Manager**.

   This will display the Bluetooth Manager dialog.
3. Tap the **Active Connections** tab.
4. Apply **power** to the **RabbitCore board**.

   Power can be applied by attaching the AC-Adapter provided by Z-World.
5. Using the Dynamic C Editor, on the **Run** menu, click **Run**.

   Depending on your current Pocket PC Bluetooth configuration, the Authorization Requested Dialog may appear (Figure 9). If this dialog appears, tap Accept to accept the connection. The Connection Status LED (Figure 1) on the eb506 module will turn on when a connection is established. On the Pocket PC the connection will be shown in the Incoming Connections section of the Active Connections tab on the Bluetooth Active Connections dialog.



Bluetooth
Authorization Requested

Service: Generic Serial

Remote Device: EB506

How to use this screen

Authorization is required to allow the above mentioned device access to the specified service.

Tap 'Accept' to allow the device to connect, or 'Deny' to reject the connection request.

Figure 9: Pocket PC Bluetooth Authorization Request Dialog

Communications

This section contains a number of exercises that demonstrate methods of communicating over a Bluetooth wireless connection with the eb506. The scenarios described are not meant to form an exhaustive list, but rather illustrate a number of more common and useful configurations. All source code shown in these exercises is available in electronic form on the accompanying CD, in the Samples folder, using the filename used in this manual.

Communicating between Two RCM3110 Modules

In this exercise we will step through the process of communicating wirelessly between two RCM3110 modules connected to Z-World prototyping boards. This communication will be used to transmit the state of the switches present on each prototyping board and use this status to control the LEDs on the remote platform.

To perform this exercise as documented, you will need two RCM3110 modules, two prototyping boards, and two eb506 modules. If you are using any other combination of hardware you may need to make adjustments to this exercise.

Step 1: Create a Callee Application for RabbitCore #1

In this step we will create a Dynamic C application that will monitor the state of the switches and transmit switch press messages to a remote eb506. It will also respond to switch messages from the remote eb506 and control its own LEDs accordingly.

1. Open the **Dynamic C Editor**.
2. Enter the following **program code** into the editor. This application is available in electronic form on the accompanying CD, in the Samples folder, in the file LEDandSwitchCallee.c.

```
auto char szPortFData;
auto int nDataAvailableOnF, nSwitchS2, nSwitchS3;

// Open the serial port for communications and initialize the eb506
serFopen(9600);
```

```
a7_serFeb506Init();

// Initialize the LEDs
WrPortI(PGDDR, &PGDDRShadow, 0xC0 | PGDDRShadow);
BitWrPortI(PGDR, &PGDRShadow, 1, LED_DS1);
BitWrPortI(PGDR, &PGDRShadow, 1, LED_DS2);

while(1)
{
   // Read data from eb506 and update the LEDs
   nDataAvailableOnF = serFrdUsed();
   if(nDataAvailableOnF > 0)
   {
      serFread(&szPortFData, 1, 10);
      switch(szPortFData)
      {
      case '0':
         BitWrPortI(PGDR, &PGDRShadow, 0, LED_DS1);
         break;
      case '1':
         BitWrPortI(PGDR, &PGDRShadow, 1, LED_DS1);
         break;
      case '2':
         BitWrPortI(PGDR, &PGDRShadow, 0, LED_DS2);
         break;
      case '3':
         BitWrPortI(PGDR, &PGDRShadow, 1, LED_DS2);
         break;
      }
   }

   // Check for changes in the switch state
   if(BitRdPortI(PGDR, SWITCH_S2) ^ nSwitchS2)
   {
      nSwitchS2 = BitRdPortI(PGDR, SWITCH_S2);
      if(!nSwitchS2)
         serFputc('0');
      else
         serFputc('1');
   }
   if(BitRdPortI(PGDR, SWITCH_S3) ^ nSwitchS3)
   {
      nSwitchS3 = BitRdPortI(PGDR, SWITCH_S3);
      if(!nSwitchS3)
         serFputc('2');
      else
         serFputc('3');
   }
}
serFclose();
```

3. On the **File** menu, click **Save As**.
4. In the **File name** box, enter a file name to which to save the program just created. For example, LEDandSwitchCallee.c.
5. Click **Save**.

Step 2: Create a Caller Application for RabbitCore #2

In this step we will create a Dynamic C application that will establish a Bluetooth connection, monitor the state of the switches, and transmit switch press messages to a remote eb506. It will also respond to switch messages from the remote eb506 and control its own LEDs accordingly.

1. Open the **Dynamic C Editor**.
2. Enter the following **program code** into the editor. This application is available in electronic form on the accompanying CD, in the Samples folder, in the file LEDandSwitchCaller.c.

```
auto char szPortFData;
auto int nDataAvailableOnF, nSwitchS2, nSwitchS3;


// Open the serial port for communications and initialize the eb506
serFopen(9600);
```

```
a7_serFeb506Init();

// Initialize the LEDs
WrPortI(PGDDR, &PGDDRShadow, 0xC0 | PGDDRShadow);
BitWrPortI(PGDR, &PGDRShadow, 1, LED_DS1);
BitWrPortI(PGDR, &PGDRShadow, 1, LED_DS2);

// Connect to the remote device
serFputs("con 00:0C:84:00:05:29\r");
a7_serFwaitForString("ACK\r", 0);

// Wait for the connection to be established and switch into data mode.
while(!BitRdPortI(PDDR, 4));
BitWrPortI(PDDR, &PDDRShadow, 1, 5);
a7_pauseMs(300);

while(1)
{
   // Read data from eb506 and update the LEDs
   nDataAvailableOnF = serFrdUsed();
   if(nDataAvailableOnF > 0)
   {
      serFread(&szPortFData, 1, 10);
      switch(szPortFData)
      {
      case '0':
         BitWrPortI(PGDR, &PGDRShadow, 0, LED_DS1);
         break;
      case '1':
         BitWrPortI(PGDR, &PGDRShadow, 1, LED_DS1);
         break;
      case '2':
         BitWrPortI(PGDR, &PGDRShadow, 0, LED_DS2);
         break;
      case '3':
            BitWrPortI(PGDR, &PGDRShadow, 1, LED_DS2);
            break;
         }
      }

      // Check for changes in the switch state
      if(BitRdPortI(PGDR, SWITCH_S2) ^ nSwitchS2)
      {
         nSwitchS2 = BitRdPortI(PGDR, SWITCH_S2);
         if(!nSwitchS2)
            serFputc('0');
         else
            serFputc('1');
      }
      if(BitRdPortI(PGDR, SWITCH_S3) ^ nSwitchS3)
      {
         nSwitchS3 = BitRdPortI(PGDR, SWITCH_S3);
         if(!nSwitchS3)
            serFputc('2');
         else
            serFputc('3');
      }
   }
   serFclose();
```

3. On the **File** menu, click **Save As**.
4. In the **File name** box, enter a file name to which to save the program just created. For example, LEDandSwitchCaller.c.
5. Click **Save**.

Step 3: Download the Applications to the Rabbit Core Modules

In this step we will download the applications we just created to the respective RabbitCore boards.

1. Select the **LEDandSwitchCallee.c** file in the Dynamic C Editor.
2. Connect the **serial port on RabbitCore #1** to the **PC**.

3. Apply **power** to **RabbitCore #1**.
4. On the **Compile** menu, click **Compile to Target->Compile to Flash**
5. Disconnect the **power** from the **RabbitCore #1**.
6. Disconnect the **serial port** from **RabbitCore #1**.
7. Select the **LEDandSwitchCaller.c** file in the Dynamic C Editor.
8. Connect the **serial port on RabbitCore #2** to the **PC**.
9. Apply **power** to **RabbitCore #2**.
10. On the **Compile** menu, click **Compile to Target->Compile to Flash**
11. Disconnect the **power** from the **RabbitCore #2**.
12. Disconnect the **serial port** from **RabbitCore #2**.

Step 4: Run the Callee / Caller Applications

In this step we will run the Callee and Caller applications.

1. Apply **power** to both of the **RabbitCore Prototyping boards**.

   After a few seconds the connection status LEDs on both eb506 boards should turn on. These LEDs indicate that there is an active connection between the two boards.

2. Press the **buttons** S2 and S3 and observe the **LEDs** DS1 and DS2.

   As you press the buttons S2 and S3 on one device you will notice the LEDs of the other device reflecting their state. When a button is depressed the corresponding LED will be lit. When a button is released, the corresponding LED will not be lit.

Communicating between a PC with DBT-120 and an RCM3110

In this exercise we will step through the process of communicating between a PC that has a D-Link® DBT-120 Bluetooth USB Adapter and an eb506 module inserted into an RCM3110 module and prototyping board.

To perform this exercise, as documented, you will need a D-Link DBT-120, an RCM3110, a prototyping board, and an eb506 module. If you are using any of the other supported Z-World boards, you may need to make adjustments to this exercise.

On the PC, the DBT-120 Bluetooth Software associates a COM port for establishing a connection from the PC to a remote Bluetooth device and a separate COM port for connections that are established from a remote Bluetooth device to the PC. The Establishing a Connection section of this manual describes how to establish the connection between devices. This exercise demonstrates how to communicate data between the PC and a remote eb506.

The D-Link DBT-120 Bluetooth USB Adapter software must be fully installed prior to establishing a connection. The PC settings shown in this exercise are based upon the software provided with the D-Link DBT-120 Bluetooth USB Adapter.

Step 1: Transmit Data from the PC to the RCM3110

In this step we will create a Dynamic C application to read data from the eb506 and display the data in the Dynamic C Editor Debug window. We will then download and run the application.

1. Connect the **RCM3110 serial port** to the **PC**.
2. Open the **Dynamic C Editor**.
3. Enter the following **program code** into the editor. This application is available in electronic form on the accompanying CD, in the Samples folder, in the file Receive.c.

```
auto char szPortFData;
auto int nDataAvailableOnF;

// Open the serial port for communications and initialize the eb506
serFopen(9600);
a7_serFeb506Init();

while(1)
{
   // Read data from eb506 and echo it to the debug port
```

```
      nDataAvailableOnF = serFrdUsed();
      if(nDataAvailableOnF > 0)
      {
         serFread(&szPortFData, 1, 10);
         printf("%c", szPortFData);
      }
   }
   serFclose();
```

The application waits for an individual byte of data to arrive and then displays the byte in the debug window and then repeats this process.

4. On the **File** menu, click **Save As**.
5. In the **File name** box, enter a file name to which to save the program just created. For example, Receive.c.
6. Click **Save**.
7. Apply **power** to the **RCM3110 board**.
8. On the **Run** menu, click **Run** to compile.

   This will display the Download Progress dialog while downloading the program to the RCM3110. After the download is complete, the Stdio dialog should be shown. If it is not, select Debug Windows from the Window menu of the Dynamic C Editor and click Stdio.
9. Establish a connection from the **PC** to the **RCM3110**.

   Please see the section titled Connecting a PC with a DBT-120 to an RCM3110 for information on establishing a connection.
10. Using HyperTerminal, type a series of **characters**.

    These characters will be transmitted over the wireless link, read by the Dynamic C application, and then displayed in the Stdio debug window.

Step 2: Transmit Data from the RCM3110 to the PC

In this step we will create a Dynamic C application to transmit data from the eb506 to the PC where we will use HyperTerminal to display the data received by the DBT-120 Bluetooth USB Adapter.

1. Recycle the power on the **eb506** attached to the **RCM3110 board** to place the eb506 into command mode.

To recycle the power on the eb506 attached to the RabbitCore board, disconnect the power, wait a couple of seconds, and then reconnect the power. The Reset push button on the prototyping board will NOT reset the eb506.

2. Close **HyperTerminal**.
3. Close the **Dynamic C Editor Debug** dialog.
4. Open **My Bluetooth Places** by double-clicking on the desktop icon.

   This will display the My Bluetooth Places dialog.
5. Click **View or modify configuration**.

   This will display the Bluetooth Configuration dialog.
6. Select the **Local Services** tab and note the **COM Port** for the Bluetooth Serial Port service. You may have to scroll to the right to see the COM Port column of the table.

   This COM port is the serial communications port that the DBT-120 Bluetooth software has associated for connections that are established from a remote Bluetooth device. This COM port can be used to communicate with the eb506 from applications, such as HyperTerminal, when connections are established from remote Bluetooth devices to the PC.
7. Select the **Hardware** tab and note the **Device Address** shown in the Device Properties section of the dialog.

   The device address will be used in the Dynamic C application developed in later actions.
8. On the **Bluetooth Configuration** dialog, click **Cancel**.

   This will close the Bluetooth Configuration dialog.
9. Close the **My Bluetooth Places** window.
10. Open **HyperTerminal**.

    This will display the Connection Description dialog.
11. In the **Name** box, type the name of your connection. For example, eb506.
12. Click **OK**.

    This will display the Connect To dialog.
13. In the **Connect using** dropdown, select the **serial port** to which the DBT-120 Bluetooth software associated with the connection from the eb506 on the RCM3110 board to the PC.

This is the COM port that we previously noted as being the COM port that is used to communicate with the eb506 when connections are established from remote Bluetooth devices to the PC.

14. Click **OK**.

    This will display the Properties dialog.

15. In the **Bits per second** dropdown, select **9600**.
16. In the **Data bits** dropdown, select **8**.
17. In the **Parity** dropdown, select **None**.
18. In the **Stop bits** dropdown, select **1**.
19. In the **Flow control** dropdown, select **None**.
20. Click **OK**.

    This will ONLY open the COM port for the DBT-120 Bluetooth USB Adapter software. This does NOT establish a connection with the eb506 on the RabbitCore. In the next steps, we will write a Dynamic C application to run on the RabbitCore which will connect the eb506 to the DBT-120 and provide the serial connection over Bluetooth.

21. On the **Call** menu, click **Disconnect**.

    This will disconnect the connection just established, so that we can modify the connection properties in the following actions.

22. On the **File** menu, click **Properties**.

    This will display the Properties dialog.

23. On the **Settings** tab, click **ASCII Setup**.

    This will display the ASCII Setup dialog.

24. Check the **Send line ends with line feeds** checkbox.
25. Check the **Echo typed characters locally** checkbox.
26. Check the **Append line feeds to incoming line ends** checkbox.
27. Check the **Wrap lines that exceed terminal width** checkbox.
28. Click **OK**.

    This will return to the Properties dialog.

29. Click **OK**.
30. On the **Call** menu, click **Call**.

This will establish a connection with the DBT-120 Bluetooth USB Adapter Software.

31. Using the Dynamic C Editor, on the **File** menu, click **New**.

    This will create a new project window within the Dynamic C Editor.

32. Enter the following **program code** into the editor, replacing the device Bluetooth address with the device address obtained from the Hardware tab of the Device Properties section of the Bluetooth Configuration dialog on the PC. This application is available in electronic form on the accompanying CD, in the Samples folder, in the file HelloWorld.c.

```
auto int nCounter;


serFopen(9600);
a7_serFeb506Init();


// Connect to the PC
serFputs("con 00:0C:84:00:05:30\r");
a7_serFwaitForString("ACK\r", 0);


// Wait for the connection to be established and switch into data mode.
while(!BitRdPortI(PDDR, 4));
BitWrPortI(PDDR, &PDDRShadow, 1, 5);
a7_pauseMs(300);


printf("Connection established\n");


// Send "Hello World" ten times
for(nCounter = 0; nCounter < 10; nCounter++)
{
   serFputs("Hello World\r");
   a7_pauseMs(1000);
}


// Verify all data has been transmitted and switch to Command Mode
while(serFwrFree() != FOUTBUFSIZE);
BitWrPortI(PDDR, &PDDRShadow, 0, 5);
```

```
a7_serFwaitForString("\r>", 0);


// Disconnect from the remote device
serFputs("dis\r");
a7_serFwaitForString("\r>", 0);
serFclose();
```

The Dynamic C application establishes a Bluetooth connection with the PC, transmits “Hello World” ten times, switches back to command mode, and then disconnects from the remote device.

The first call to serFputs is used when the eb506 is in command mode and instructs the eb506 to establish a connection with the device specified. Once a connection is established the eb506 is switched to data mode, which causes further calls to serFputs to be sent to the remote device.

33. On the **File** menu, click **Save As**.

    This will display the Save As dialog.

34. In the **File name** box, enter a file name to which to save the program just created. For example, HelloWorld.c.
35. Click **Save**.
36. On the **Run** menu, click **Run**.

    This will display the Download Program dialog while downloading the program to the RabbitCore module. After the download is complete the Dynamic C application will transmit “Hello World” over the wireless link, and HyperTerminal will display the received data.

Communicating between a PC with XP SP2 and an RCM3110

In this exercise we will step through the process of communicating between a PC that has a Bluetooth USB adapter that is running Windows XP SP2 and an eb506 module connected to an RCM3110 module and Z-World prototyping board.

To perform this exercise, as documented, you will need a PC running Windows XP SP2, a Bluetooth USB adapter, a Z-World RCM3110 module, a Z-World RabbitCore prototyping board, and an eb506 module. If you are using any of the other supported Z-World boards, you may need to make adjustments to this exercise.

On the PC, the Microsoft Bluetooth Software associates a COM port for establishing a connection from the PC to a remote Bluetooth device and a separate COM port for connections that are established from a remote Bluetooth device to the PC. The Establishing a Connection section of this manual describes how to establish the connection between devices. This exercise demonstrates how to communicate data between the PC and a remote eb506.

Step 1: Transmit Data from the PC to the RCM3110

In this step we will create a Dynamic C application to read data from the eb506 and display the data in the Dynamic C Editor Debug window. We will then download and run the application.

1. Connect the **RCM3110 serial port** to the **PC**.
2. Open the **Dynamic C Editor**.
3. Enter the following **program code** into the editor. This application is available in electronic form on the accompanying CD, in the Samples folder, in the file Receive.c.

```
auto char szPortFData;
auto int nDataAvailableOnF;

// Open the serial port for communications and initialize the eb506
serFopen(9600);
a7_serFeb506Init();

while(1)
{
   // Read data from eb506 and echo it to the debug port
   nDataAvailableOnF = serFrdUsed();
   if(nDataAvailableOnF > 0)
```

```
      {
         serFread(&szPortFData, 1, 10);
         printf("%c", szPortFData);
      }
   }
   serFclose();
```

The application waits for an individual byte of data to arrive and then displays the byte in the debug window and then repeats this process.

4. On the **File** menu, click **Save As**.
5. In the **File name** box, enter a file name to which to save the program just created. For example, Receive.c.
6. Click **Save**.
7. Apply **power** to the **RCM3110 board**.
8. On the **Run** menu, click **Run** to compile.

   This will display the Download Progress dialog while downloading the program to the RCM3110. After the download is complete, the stdio dialog should be shown. If it is not, select Debug Windows – stdio from the Window menu of the Dynamic C Editor.
9. Establish a connection from the **PC** to the **RCM3110**.

   Please see the section titled Connecting a PC with XP SP2 for information on establishing a connection.
10. Using HyperTerminal, type a series of **characters**.

    These characters will be transmitted over the wireless link, read by the Dynamic C application, and then displayed in the stdio debug window.

Step 2: Transmit Data from the RCM3110 to the PC

In this step we will create a Dynamic C application to transmit data from the eb506 to the PC where we will use HyperTerminal to display the data received.

1. Open **HyperTerminal** on the port for inbound Bluetooth connections.

   Please see the section titled Communicating between a PC with DBT-120 and an RCM3110 for information on configuring HyperTerminal for the inbound Bluetooth port.
2. Using the Dynamic C Editor, on the **File** menu, click **New**.

   This will create a new project window within the Dynamic C Editor.

3. Enter the following **program code** into the editor, replacing the device Bluetooth address with the device address obtained from the Hardware tab of the Device Properties section of the Bluetooth Configuration dialog on the PC. This application is available in electronic form on the accompanying CD in the Samples folder in the file HelloWorld.c.

```
auto int nCounter;

serFopen(9600);
a7_serFeb506Init();

// Connect to the PC
serFputs("con 00:0C:84:00:05:30\r");
a7_serFwaitForString("ACK\r", 0);

// Wait for the connection to be established and switch into data mode.
while(!BitRdPortI(PDDR, 4));
BitWrPortI(PDDR, &PDDRShadow, 1, 5);
a7_pauseMs(300);

printf("Connection established\n");

// Send "Hello World" ten times
for(nCounter = 0; nCounter < 10; nCounter++)
{
   serFputs("Hello World\r");
   a7_pauseMs(1000);
}

// Verify all data has been transmitted and switch to Command Mode
while(serFwrFree() != FOUTBUFSIZE);
BitWrPortI(PDDR, &PDDRShadow, 0, 5);
a7_serFwaitForString("\r>", 0);

// Disconnect from the remote device
serFputs("dis\r");
```

```
a7_serFwaitForString("\r>", 0);
serFclose();
```

The Dynamic C application establishes a Bluetooth connection with the PC, transmits "Hello World" ten times, switches back to command mode, and then disconnects from the remote device.

The first call to serFputs is used when the eb506 is in command mode and instructs the eb506 to establish a connection with the device specified. Once a connection is established the eb506 is switched to data mode, which causes further calls to serFputs to be sent to the remote device.

4. On the **File** menu, click **Save As**.

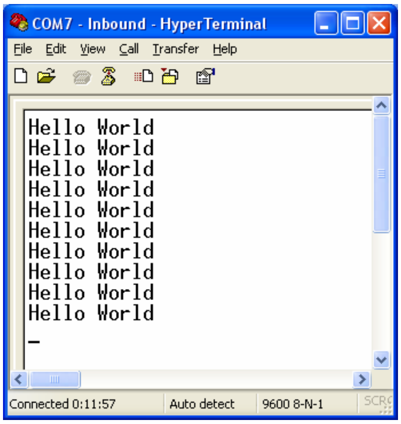
   This will display the Save As dialog.

5. In the **File name** box, enter a file name to which to save the program just created. For example, HelloWorld.c.

6. Click **Save**.

7. On the **Run** menu, click **Run**.

   This will display the Download Program dialog while downloading the program to the RabbitCore module. After the download is complete the Dynamic C application will transmit "Hello World" over the wireless link, and HyperTerminal will display the received data (Figure 10).


COM7 - Inbound - HyperTerminal
File Edit View Call Transfer Help
Connected 0:11:57
Auto detect
9600 8-N-1
SCR

```
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
_
```

Figure 10: HyperTerminal application with data received from the eb506/RabbitCore

Communicating between a Pocket PC 2003 and an RCM 3110

In this exercise we will step through the process of communicating between a Pocket PC 2003 device with integrated Bluetooth, and an eb506 module inserted into a RCM3110 board.

To perform this exercise, as documented, you will need a Pocket PC 2003 device, an RCM3110 board, and an eb506 module. Depending on the specific Pocket PC model that you are using, you may need to make minor adjustments to this exercise.

Step 1: Transmit Data from the Pocket PC to the RCM3110

In this step we will create a Dynamic C application to read data from the eb506 and display the data in the Dynamic C Editor Debug window. We will then download and run the application. The application for the Pocket PC is too verbose to include in this manual; therefore, the application, along with the source code, is available on the accompanying CD, in the Samples folder. To modify the Pocket PC application you will need eMbedded Visual C++ 4.0 with Service Pack 2 and the SDK for Windows Mobile™ 2003-based Pocket PCs.

1. Connect the **RCM3110 module serial port** to the **PC**.
2. Open the **Dynamic C Editor**.
3. Enter the following **program code** into the editor. This application is available in electronic form on the accompanying CD, in the Samples folder, in the file Receive.c.

```
auto char szPortFData;
auto int nDataAvailableOnF;

// Open the serial port for communications and initialize the eb506
serFopen(9600);
a7_serFeb506Init();

while(1)
{
   // Read data from eb506 and echo it to the debug port
   nDataAvailableOnF = serFrdUsed();
   if(nDataAvailableOnF > 0)
   {
      serFread(&szPortFData, 1, 10);
      printf("%c", szPortFData);
   }
}
serFclose();
```

4. On the **File** menu, click **Save As**.
5. In the **File name** box, enter a file name to which to save the program just created. For example, Receive.c.
6. Click **Save**.
7. Apply **power** to the **RCM3110 board**.
8. On the **Run** menu, click **Run** to compile.

   This will display the Download Progress dialog while downloading the program to the RCM3110. After the download is complete, the stdio dialog should be shown. If it is not, select Debug Windows – stdio from the Window menu of the Dynamic C Editor.
9. On the Pocket PC, tap the **Bluetooth icon** in the system tray on the Today screen and select **Bluetooth Settings**.

   This will display the Settings dialog.
10. Scroll to the right, tap the **Serial Port** tab and note the **Outbound COM port**.

    The Outbound COM port will be used in the Pocket PC application later in this step.
11. Download the **PPCTxToEB** application to the **Pocket PC 2003 device**.
12. Run the **PPCTxToEB** application (Figure 11).

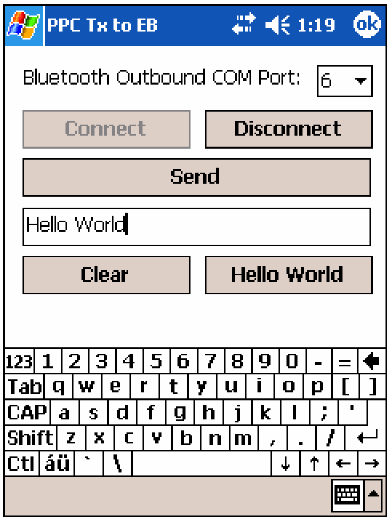


Figure 11: PPCTxToEB Pocket PC Application

13. In the **Bluetooth Outbound COM Port** dropdown, select the **COM port number** that matches the Bluetooth Outbound COM Port, which we previously discovered.
14. Tap the **Connect** button.

    This will display the Bluetooth Browser dialog (Figure 12).

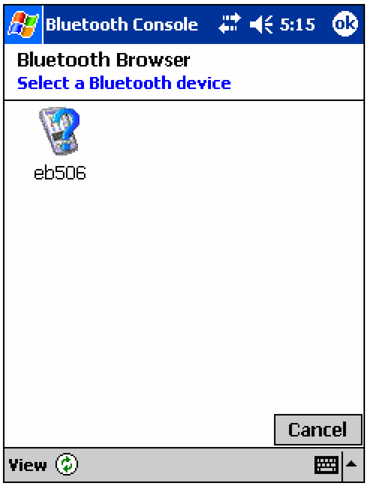


Figure 12: Pocket PC Bluetooth Browser Dialog

15. Tap **eb506** in the Bluetooth Browser dialog to establish a connection with the eb506 on the RabbitCore board.

    If there are no devices shown in the Bluetooth Browser dialog, tap the refresh icon to search for your Bluetooth device.

    A connection will be established with the device and the PPCTxToEB application will be shown again.

16. Enter some text to send and tap **Send**.

    This will transmit the ASCII text over the wireless link. The Dynamic C application will then receive these characters and display them in the Dynamic Editor Debug window.

    You can also tap the **Hello World** button to have the application fill the edit box with the text: “Hello World”.

17. Tap the **Disconnect** button to close the Bluetooth connection.

Step 2: Transmit Data from the RCM3110 to the Pocket PC

In this step we will create a Dynamic C application to transmit data from the eb506 to the Pocket PC. We will then download and run the application. The application for the Pocket PC is too verbose to include in this manual; therefore, the application, along with the source code, is available on the accompanying CD in the Samples folder. To modify the Pocket PC application you will need eMbedded Visual C++ 4.0 with Service Pack 2 and the SDK for Windows Mobile™ 2003-based Pocket PCs.

1. Using the Dynamic C Editor, on the **File** menu, click **New**.

   This will create a new project window within the Dynamic C Editor.

2. Enter the following **program code** into the editor, replacing the device address with the device address obtained from the Pocket PC. This application is available in electronic form on the accompanying CD, in the Samples folder, in the file HelloWorld.c.

```
auto int nCounter;


serFopen(BAUD_RATE);

a7_serFeb506Init();


// Connect to the PPC

serFputs("con 00:0C:84:00:05:31\r");

a7_serFwaitForString("ACK\r", 0);
```

```
// Wait for the connection to be established and switch into data mode.
while(!BitRdPortI(PDDR, 4));
BitWrPortI(PDDR, &PDDRShadow, 1, 5);
a7_pauseMs(300);

printf("Connection established\n");

// Send "Hello World" ten times
for(nCounter = 0; nCounter < 10; nCounter++)
{
   serFputs("Hello World\r");
   a7_pauseMs(1000);
}

// Verify all data has been transmitted and switch to Command Mode
while(serFwrFree() != FOUTBUFSIZE);
BitWrPortI(PDDR, &PDDRShadow, 0, 5);
a7_serFwaitForString("\r>", 0);

// Disconnect from the remote device
serFputs("dis\r");
a7_serFwaitForString("\r>", 0);
serFclose();
```

The Dynamic C application establishes a connection with the Pocket PC, transmits “Hello World” ten times, switches back to command mode, and then disconnects from the remote device.

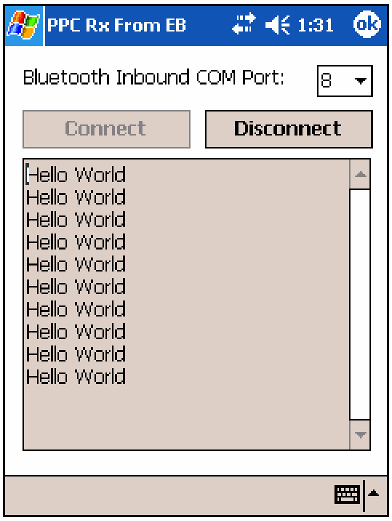
The first call to serFputs is used when the eb506 is in command mode and instructs the eb506 to establish a connection with the device specified. Once a connection is established the eb506 switches to data mode, which causes further calls to serFputs to be sent to the remote device.

3. On the **File** menu, click **Save As**.
4. In the **File name** box, enter a file name to which to save the program just created. For Example, HelloWorld.c.

5. Click **Save**.
6. On the Pocket PC, tap the **Bluetooth icon** in the system tray on the Today screen and select **Bluetooth Settings**.

   This will display the Settings dialog.
7. Scroll to the right, tap the **Serial Port** tab and note the **Inbound COM port**.

   The Inbound COM port will be used in the Pocket PC application later in this step.
8. Download the **PPCRxFromEB** application to the **Pocket PC**.
9. Run the **PPCRxFromEB** application (Figure 13).



Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World

Figure 13: PPCRxFromEB Pocket PC Application

10. In the **Bluetooth Inbound COM Port** dropdown, select the **COM port number** that matches the Bluetooth Inbound COM Port, which we discovered in a previous action.
11. Tap the **Connect** button.
12. Apply **power** to the RabbitCore board.
13. Using the Dynamic C Editor, on the **Run** menu, click **Run**.

    This will display the Download Progress dialog while downloading the program to the RCM3110. After the download is complete the Dynamic application will establish a connection with the Pocket PC. Depending on your current Pocket PC Bluetooth configuration, the Authorization Request Dialog may appear (Figure 9). If this dialog appears, tap Accept the connection. Once the connection is established, the Dynamic C application will transmit “Hello World” over the wireless link (10 times), and the Pocket PC application will display the received data in the window.

14. On the Pocket PC, tap the **Disconnect** button to close the connection.

This page intentionally left blank.

Security

This section contains a number of exercises that demonstrate various security scenarios that can be implemented when using the eb506 module. The scenarios described are not meant to form an exhaustive list, but rather illustrate a number of more common and useful configurations. All source code shown in these exercises is available in electronic form on the accompanying CD, in the Samples folder, using the filename used in this manual.

Strong Security on an RCM3110 Module

Cellular phones are typically configured for strong Bluetooth security because not only do they allow you to make phone calls, but they often contain sensitive contact information as well. In this exercise we will demonstrate how to use the eb506 module to implement the strong Bluetooth security model used on cellular phones.

To perform this exercise, as documented, you will need a PC running Windows XP SP2, a Bluetooth USB adapter, a Z-World RCM3110 module, a Z-World RabbitCore prototyping board, and an eb506 module. If you are using any of the other supported Z-World boards, you may need to make adjustments to this exercise.

Step 1: Configure Bluetooth security through a Dynamic C application

In this step we will write a Dynamic C application that allows you to change security settings by using the switch, S2, on the prototyping board. We will then download and run the application.

Note about passkeys: Passkeys should be created with the same idea in mind as creating a password for your own personal use. Some tips include using at least eight characters, including a combination of uppercase letters, lowercase letters, symbols, and numbers. The more characters and different types of characters you use, the more secure your passkey will be. Additionally, it’s recommended not to use words that can be found in the dictionary as this allows for a common dictionary attack to easily crack your passkey. This Dynamic C application changes the passkey of the eb506 module. You will need to make note of the passkey as it will be needed in the next step when we connect the PC to the eb506.

1. Connect the **RCM3110 serial port** to the **PC**.
2. Open the **Dynamic C Editor**.
3. Enter the following **program code** into the editor. This application is available in electronic form on the accompanying CD in the Samples folder in the file Security.c.

```
#define LED_DS1 6 // LED, port G bit 6
#define LED_DS2 7 // LED, port G bit 7
#define SWITCH_S2 1 // Switch, port G bit 1
#define SWITCH_S3 0 // Switch, port G bit 0

int SetSecurityClosed()
{
  // Set visible off, security closed
  serFputs("set visible off\r");
  a7_serFwaitForString("ACK\r>", 0);
  serFputs("set security closed\r");
  a7_serFwaitForString("ACK\r>", 0);

  // Turn the LED off and return
  BitWrPortI(PGDR, &PGDRShadow, 1, LED_DS1);
  return 0;
}

int SetSecurityOpen()
{
  // Set visible on, security open
  serFputs("set visible on\r");
  a7_serFwaitForString("ACK\r>", 0);
  serFputs("set security open\r");
  a7_serFwaitForString("ACK\r>", 0);
  return 1;
}

void main()
{
```

```
auto int nSwitchS2, nLEDDS1, nSecurity, nConnected, nBlinkRate;
auto unsigned long lStartTick;

nLEDDS1 = 1;        // LED is off
nBlinkRate = 500;   // Half second
nConnected = 0;     // 0 = NOT connected, 1 = connected
nSecurity = 0;      // 0 = visible off, security closed
                    // 1 = visible on, security open

// Open the serial port for communications and initialize the eb506
serFopen(9600);
a7_serFeb506Init();

// Initialize the LEDs
WrPortI(PGDDR, &PGDDRShadow, 0xC0 | PGDDRShadow);
BitWrPortI(PGDR, &PGDRShadow, 1, LED_DS1);
BitWrPortI(PGDR, &PGDRShadow, 1, LED_DS2);

// Initialize the switch
nSwitchS2 = BitRdPortI(PGDR, SWITCH_S2);

// Use a secure passkey
serFputs("set passkey a7blue*506\r");
a7_serFwaitForString("ACK\r>", 0);

// Use encryption
serFputs("set encrypt on\r");
a7_serFwaitForString("ACK\r>", 0);

// Start with security closed
nSecurity = SetSecurityClosed();

while(1)
{
  // Check for changes in the switch state
```

```
if(BitRdPortI(PGDR, SWITCH_S2) ^ nSwitchS2)
{
  nSwitchS2 = BitRdPortI(PGDR, SWITCH_S2);

  // Toggle the security mode when the switch is pressed
  if(nSwitchS2 && nConnected == 0)
  {
    if(nSecurity == 0)
    {
      nSecurity = SetSecurityOpen();

      // Setup for making the LED blink
      nLEDDS1 = 0;
      lStartTick = MS_TIMER;
    }
    else
    {
      nSecurity = SetSecurityClosed();
    }
  }
}

if(nSecurity == 1 && nConnected == 0)
{
  // Make the LED blink to show we are have
  // visible on, security open
  if((MS_TIMER - lStartTick) < nBlinkRate)
  {
    if(nLEDDS1 == 0)
      nLEDDS1 = 1;
    else
      nLEDDS1 = 0;

    lStartTick += nBlinkRate;
  }
```

```
      BitWrPortI(PGDR, &PGDRShadow, nLEDDS1, LED_DS1);
    }

    // Ensure the prototyping board LED, DS1, mimics
    // the connection LED of the eb506
    if(BitRdPortI(PDDR, 4))
    {
      if(nConnected == 0)
      {
        // State change from disconnected to connected
        nConnected = 1;
        BitWrPortI(PGDR, &PGDRShadow, 0, LED_DS1);
      }
    }
    else
    {
      if(nConnected == 1)
      {
        // State change from connected to disconnected
        nConnected = 0;
        BitWrPortI(PGDR, &PGDRShadow, 1, LED_DS1);
        nSecurity = SetSecurityClosed();
      }
    }
  }

  serFclose();
}
```

The application waits for the switch to be pressed. Once the switch is pressed, the eb506 module goes into a more vulnerable mode; it becomes visible and switches to open security enabling new trusted relationships. When in open security mode, the LED, DS1, will blink. By pressing the switch again, the eb506 will return to closed security mode and visibility will be turned off.

4. On the **File** menu, click **Save As**.
5. In the **File name** box, enter a file name to which to save the program just created. For example, Security.c.
6. Click **Save**.
7. Apply **power** to the **RCM3110 board**.
8. On the **Run** menu, click **Run** to compile.

   This will display the Download Progress dialog while downloading the program to the RCM3110.

   A device that implements strong security is most vulnerable when allowing devices that have not yet been trusted to connect, verify their passkey, and become trusted. There is nothing inherently unsafe about this process; however it is a time when intruders have the most opportunity to compromise the device.

   There are two main concerns when security is set to open. The first is that if a weak passkey is set, an intruder may be able to guess the passkey and therefore gain access to the device. This can be largely avoided however, by first choosing a strong passkey and second by only placing the device into open security mode temporarily in a safe environment and then returning to closed security mode.

   The second concern is that an intruder pulls sensitive information out of the air that either is directly useful or allows trusted access to the device. This is only possible during a small window of time when an untrusted device is first connected and becomes trusted, so this process should be carried out in a safe environment if possible. Once a device is considered trusted, no information that could be used by an intruder to gain access to the device is transmitted over the air. If encryption is enabled as well, then no information including the data sent over the air is vulnerable to an intruder. By default EmbeddedBlue modules use 56-bit encryption for all data when enabled.

Step 2: Connecting a PC with XP SP2 to an RCM3110 in open security

In this step, we will run the application we wrote in the previous section, which initially runs in closed security mode with visibility set to off. The switch, S2, can be pressed to change into open security mode with visibility on. At this point, connect your PC with XP SP2 to the remote Bluetooth device and become “trusted.” After returning to closed security mode with visibility turned off, only trusted devices (the PC) can connect to the remote Bluetooth device.

1. **Disconnect** the **serial cable** from the RabbitCore.

2. **Recycle** the power on the prototyping board by unplugging the power cable and plugging the cable in.

3. **Press** the switch, **S2**, on the prototyping board.

   This puts the eb506 module in open security mode with visibility on. Note the blinking LED which informs us we are vulnerable to other devices becoming trusted with the eb506 if they have the correct passkey.

4. **Connect** your PC with XP SP2 to the remote **Bluetooth device**.

   See the section named Connecting a PC with XP SP2 to an RCM3110 for more information. Upon successful connection, the PC and eb506 will establish a trusted relationship. You will need to the passkey from the Dynamic C application written in the previous section. The Dynamic C application changed the passkey to a more secure passkey from the default of 0000 by using the command set passkey.

5. **Disconnect** the PC and the remote Bluetooth device.

6. **Press** the switch, **S2**, to return to closed security with visibility off.

   Notice the prototyping board LED will be off.

7. **Connect** to the **eb506** module from the PC with XP SP2, referring to the section named Connecting a PC with XP SP2 to an RCM3110 for more information.

   The PC can now connect and disconnect with the remote Bluetooth device in a secure manner since visibility is now off and security is closed. Remember that when security is closed, no other devices are allowed to become "trusted."

   The LED on the prototyping board will mimic the connection LED on the eb506 module. When there is a connection between the PC and the remote Bluetooth device, the LED on the prototyping board, DS1, will be on. Likewise, it will be off when there is no connection.

   By pressing the switch, S2, the eb506 module will go back into open security mode with visibility set to on. At this point, other devices can become trusted.

   Note that when the eb506 is connected, the switch, S2, does NOT toggle between the security settings.

This page intentionally left blank.

Command Set

The EmbeddedBlue command set is comprised of visible ASCII characters. Therefore, a command can be issued from a terminal application, such as HyperTerminal, or directly from a custom application program, written in a programming language such as C++ or Visual Basic, running on a PC, using a RabbitCore module configured for pass-through mode. From a Dynamic C application, these commands can be issued by using the serXputs and serXgets commands.

Command Basics

Commands may only be sent to the module when it is in Command Mode. White spaces are used to separate parameters of the command and a carriage-return is used to mark the end of the command. Upon receipt of a command the eb506 begins to parse the parameters. If the syntax of the command is correct the eb506 returns an ACK string, not the ACK character (0x06); otherwise, a NAK string is returned. Following the ACK or NAK string is a carriage-return (0x0D) character. If an error occurs while processing the command an error string is returned followed by a carriage-return followed by the prompt (>) character. If the command executed successfully the module will issue the prompt (>) character. Please see the Error Codes section for a description of the error codes.

The following example shows the basic structure of a command. A prompt (>) is issued by the EmbeddedBlue module. A command followed by a carriage-return is sent to the module. The module responds with either an ACK or NAK string followed by a carriage-return. If an error occurs, the module responds with an Err string followed by a space followed by an ASCII string numeric value followed by a carriage-return. A prompt (>) is then issued by the module.

```
>command<CR>

ACK | NAK<CR>

Err number<CR>

>
```

Command Error Handling in Dynamic C Applications

The following Dynamic C sample issues a Connect command, waits for the `ACK<CR>` response from the module, and then examines the next character for either a prompt (`>`) on success or the beginning of an error string on failure. If an error has occurred, the code is written to the Dynamic C Editor stdio window.

```
auto char szBuffer[7];
memset(szBuffer, 0, sizeof(szBuffer));

serFopen(9600);
a7_serFeb506Init();

// Connect to the remote device.
serFputs("con 00:0C:84:00:05:29\r");
a7_serFwaitForString("ACK\r", 0);

// Either an Err #<CR> or a '>' will be received
while((szBuffer[0] = serFgetc()) == 255);
if(szBuffer[0] == 'E')
{
   // Read the rest of the error string
   while(serFread(&szBuffer[1], 5, 50) == 0);
   printf("An error occurred during connect: %s\n", szBuffer);
   return;
}
```

Connect

The *connect* command establishes a connection to another Bluetooth device. The *connect* command may be canceled before a connection is established by issuing a carriage-return to the EmbeddedBlue device. It can take up to four seconds to cancel the connection request.

Syntax

con *address* [*timeout*]<CR>

Parameters

address	The Bluetooth address of the remote device. The Bluetooth device address is the 48-bit IEEE address which is unique for each Bluetooth unit. The format of a Bluetooth device address is a series of six hexadecimal byte values separated by colons, i.e., 00:0C:84:00:05:29.
timeout	An optional parameter used to abort the connection request after the specified number of seconds. The maximum value is 120 seconds.

Example

```
>con 00:0C:84:00:05:29<CR>

ACK<CR>

>
```

Delete Trusted Device

The *delete trusted device* command removes the remote device from trusted status and prevents it from being able to connect with the EmbeddedBlue device when security is set to closed. A delete can be performed for either a single device by passing its device address or for all trusted devices by specifying the keyword all*.*

Syntax

del trusted all **|** *address*<CR >

Parameters

all	This parameter is used to remove all devices from trusted status.
address	The Bluetooth address of the device that should be removed from trusted status. The Bluetooth device address is the 48-bit IEEE address which is unique for each Bluetooth unit. The format of a Bluetooth device address is a series of six hexadecimal byte values separated by colons, i.e., 00:0C:84:00:05:29.

Example

```
>del trusted 00:0C:84:00:05:29<CR>

ACK<CR>

>
```

Disconnect

The *disconnect* command closes the connection with the remote Bluetooth device.

Syntax

dis<CR>

Example

```
>dis<CR>
ACK<CR>
>
```

Get Address

The *get address* command returns the address of the local EmbeddedBlue device.

Syntax

get address<CR>

Returns

The unique address of the local EmbeddedBlue device used to identify the module when making connections. In Bluetooth terminology this is the Bluetooth Device Address.

Example

```
>get address<CR>

ACK<CR>

00:0C:84:00:05:29<CR>

>
```

Get Connectable Mode

The *get connectable mode* command returns the connectable mode setting of the local EmbeddedBlue device.

Syntax

get connectable<CR>

Returns

The current connectable mode setting of the local EmbeddedBlue device. In Bluetooth terminology, the returned value reflects the current setting for page scan.

on	The device will accept connections.
off	The device will NOT accept connections.

Example

```
>get connectable<CR>
ACK<CR>
on<CR>
>
```

Get Encrypt Mode

The *get encrypt mode* command returns the current encryption setting of the module. This setting controls whether the module encrypts transmitted data when security is set to either open or closed. By default EmbeddedBlue modules use 56-bit encryption, but 128-bit encryption is available upon request. Contact A7 Engineering for more information about getting 128-bit encryption.

Syntax

get encrypt<CR>

Returns

The current encrypt mode setting of the module.

on	If security is set to open or closed, transmitted data will be encrypted.
off	Transmitted data will NOT be encrypted.

Example

```
>get encrypt<CR>
ACK<CR>
on<CR>
>
```

Get Escape Character

The *get escape character* command returns the current character used in the Switch to Command Mode command to instruct the EmbeddedBlue device to leave Data Mode and enter Command Mode.

Syntax

get escchar<CR>

Example

```
>get escchar<CR>
ACK<CR>
+<CR>
>
```

Get Flow Control

The *get flow control* command returns the flow control setting of the local EmbeddedBlue device.

Syntax

get flow<CR>

Returns

none	The EmbeddedBlue device is configured for no flow control and both the RTS and CTS lines are configured as high Z inputs. A Dynamic C application is free to use these lines as normal I/O.
hardware	The EmbeddedBlue device is configured for hardware flow control and the RTS line is used for receive flow control and the CTS line is used for transmit flow control.

Example

```
>get flow<CR>

ACK<CR>

none<CR>

>
```

Get Link Timeout

The *get link timeout* command returns the amount of time, in seconds, it takes for the local EmbeddedBlue device to notice that the connection has been broken if the remote device disappears. This timeout also has an effect on how robust the communications link is to interference. If this value is set very low, the link may be lost if interference picks up for several seconds, such as when a heavy burst of 802.11 traffic is encountered.

Syntax

get linktimeout<CR>

Example

```
>get linktimeout<CR>
ACK<CR>
5<CR>
>
```

Get Name

The *get name* command returns the name of the local device. This is the value that is transmitted when a remote device performs an Inquiry and then requests the device name. If you look for local Bluetooth devices from a PC or PDA, this is the value that will be displayed to the user.

Syntax

get name<CR>

Example

```
>get name<CR>

ACK<CR>

eb506<CR>

>
```

Get Security Mode

The *get security mode* command returns the modules current security mode setting. When security is turned off, the module will allow connections to be established by any Bluetooth device. When security is set to open, the remote Bluetooth device is required to provide a valid passkey before a connection can be established. When security is set to closed, only existing trusted devices are allowed to establish connections.

For maximum security it is recommended that the module be operated in closed mode whenever possible.

Note: The Security mode is not applicable if connectable mode is set to off.

Syntax

get security<CR>

Returns

The current security mode setting of the module.

off	The module will allow any Bluetooth device to establish a connection.
open	The module will allow any Bluetooth device that provides the correct passkey to establish a connection.
closed	The module will only allow trusted devices to establish a connection.

Example

```
>get security<CR>
ACK<CR>
open<CR>
>
```

Get Visible Mode

The *get visible mode* command returns the modules current visibility setting. This setting controls whether the module can be seen by other Bluetooth devices.

Syntax

get visible<CR>

Returns

The current visible mode setting of the module. In Bluetooth terminology, the returned value reflects the current setting for inquiry scan.

on	The module is visible to other devices.
off	The module is NOT visible to other devices.

Example

```
>get visible<CR>
ACK<CR>
on<CR>
>
```

Help

The *help* command returns a listing of the EmbeddedBlue commands and a brief description of each command.

Syntax

hlp [*command*]<CR>

Parameters

command	The EmbeddedBlue command name (con, del, dis, get, lst, rst, set, and ver) for which to return help.

Examples

```
>hlp<CR>
ACK<CR>
… Help Information …
<CR>
>

>hlp con<CR>
ACK<CR>
… Help Information on the Connect Command …
<CR>
>
```

List Trusted Devices

The *list trusted devices* command returns a list of all the devices that are allowed to connect when security is set to closed. The maximum number of devices that can be trusted at any given time is twenty five, so this command will return a list of between zero and twenty five addresses. When security is set to open, new devices can be added to this list by presenting the proper passkey while establishing a new connection.

Syntax

lst trusted<CR>

Returns

List of the trusted device addresses. These devices are the only ones that are allowed to connect with this module when security is set to closed.

Example

```
>lst trusted<CR>

ACK<CR>

00:0C:84:00:05:29<CR>

00:80:C8:35:2C:B8<CR>

>
```

List Visible Devices

The *list visible devices* command returns a listing of all the devices that are currently in range and visible. The command may be canceled before the timeout is reached by sending an additional carriage-return to the module.

Syntax

lst visible [*timeout*]<CR>

Parameters

timeout	An optional parameter used to abort the list request after the specified number of seconds. The default value is 30. The maximum value is 120 seconds.

Returns

The addresses of the Bluetooth devices that are in range and visible.

Example

```
>lst visible<CR>

ACK<CR>

00:0C:84:00:05:29<CR>

00:80:C8:35:2C:B8<CR>

>
```

Reset Factory Defaults

The *reset factory defaults* command restores all module settings to factory defaults. This includes the baud rate parameter which may cause serial communications to be lost after the command is issued. To reestablish communications with the module, simply adjust the baud rate on the microprocessor serial port to match the module default rate of 9600bps.

Syntax

rst factory<CR >

Parameters

factory	Resets all settings to factory defaults.

Example

```
>rst factory<CR>
ACK<CR>
>
```

Return to Data Mode

The *return to data mode* command instructs the module to enter Data Mode when there is an active connection.

Syntax

ret<CR>

Example

```
>ret<CR>

ACK<CR>

>
```

Set Baud Rate

The *set baud rate* command sets the baud rate for communications with the local EmbeddedBlue module.

Syntax

set baud *rate* [*]<CR>

Parameters

rate	The baud rate value. Valid baud rates are 9600 (default), 19200, 38400, 57600, 115200, and 230400. Once the baud rate has been set, applications, such as HyperTerminal, must also be configured to the same baud rate to continue communicating with the eb506.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set baud 19200<CR>

ACK<CR>

>
```

Set Connectable Mode

The *set connectable mode* command provides control over whether the local EmbeddedBlue module will accept connections from other Bluetooth devices. In Bluetooth terminology, this command controls the setting for page scan.

Syntax

set connectable on **|** off [*]<CR>

Parameters

on	Configures the module so that other Bluetooth devices may establish a connection.
off	Configures the module so that other Bluetooth devices may not establish a connection.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set connectable off<CR>
ACK<CR>
>
```

Set Encrypt Mode

The *set encrypt mode* command provides control over whether transmitted data is encrypted or sent in the clear. This setting is only in effect when security is set to either open or closed. When security is turned off, the transmitted data is never encrypted. By default EmbeddedBlue modules use 56-bit encryption, but 128-bit encryption is available upon request. Contact A7 Engineering for more information about getting 128-bit encryption.

Syntax

set encrypt on | off [*]<CR>

Parameters

on	Configures the module so that transmitted data will be encrypted when security is set to either open or closed.
off	Configures the module so that transmitted data will NOT be encrypted.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set encrypt on<CR>

ACK<CR>

>
```

Set Escape Character

The *set escape character* command provides control over the character used in the Switch to Command Mode command to instruct the module to leave Data Mode and enter Command Mode. The factory default escape character is the plus sign (+).

Syntax

set escchar *character* [*]<CR>

Parameters

character	The character the module should recognize as the escape character used in the Switch to Command Mode command.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set escchar & *<CR>

ACK<CR>

>
```

Set Flow Control

The *set flow control* command provides control over the flow control setting of the local EmbeddedBlue module.

Syntax

set flow none | hardware [*] <CR>

Parameters

none	Configures the module for no flow control and both the RTS and CTS lines are configured as high Z inputs. This allows a Dynamic C application to use these lines a normal I/O.
hardware	Configures the module for hardware flow control. The RTS line is used for receive flow control and the CTS line is used for transmit flow control.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set flow none<CR>

ACK<CR>

>
```

Set Link Timeout

The *set link timeout* command sets the amount of time it takes for the local EmbeddedBlue module to notice that the connection has been broken, if the remote device disappears. This timeout also has an effect on how robust the communications link is to interference. If this value is set very low, the link may be lost if interference picks up for several seconds, such as when a heavy burst of 802.11 traffic is encountered. In Bluetooth terminology, this command controls the setting for link supervisor timeout.

Syntax

set linktimeout *timeout* [*]<CR>

Parameters

timeout	The time, in seconds, it takes for the module to notice that a connection has been broken. The default value is 5. The maximum value is 40 seconds.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set linktimeout 10<CR>

ACK<CR>

>
```

Set Name

The *set name* command sets the name of the local device. This is the value that is transmitted when a remote device performs an Inquiry and then requests the device name. If you look for local Bluetooth devices from a PC or PDA, this is the value that will be displayed to the user.

Syntax

set name *value* [*]<CR>

Parameters

value	A new device name. This value can be up to 32 characters in length and may contain any valid ASCII character.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set name eb506<CR>

ACK<CR>

>
```

Set Passkey

The *set passkey* command sets the passkey that is used when establishing a connection with security set to open. The passkey is set to 0000 by default, but this value should be changed to enhance security. It is recommended that you use a passkey that is 8 to 16 digits long.

Syntax

set passkey *value* [*]<CR>

Parameters

value	A new passkey value that is between 1 and 16 digits long.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set passkey MyNewKey<CR>
ACK<CR>
>
```

Set Security Mode

The *set security mode* command sets the module's current security mode setting. When security is turned off, the module will allow connections to be established by any Bluetooth device. When security is set to open, the remote Bluetooth device is required to provide a valid passkey before a connection can be established. When security is set to closed, only existing trusted devices are allowed to establish connections.

For maximum security it is recommended that the module be set to closed mode whenever possible.

Syntax

set security off | open | closed [*]<CR>

Parameters

off	Turns security off allowing any Bluetooth device to establish a connection.
open	Configures the module to require other devices to provide the correct passkey before establishing a connection.
closed	Configures the module to only allow trusted devices to establish a connection.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set security open<CR>

ACK<CR>

>
```

Set Visible Mode

The *set visible mode* command provides control over whether the module can be seen by other Bluetooth devices. In Bluetooth terminology, this command controls the setting for inquiry scan.

Syntax

set visible on **|** off [*]<CR>

Parameters

on	Configures the module so that other Bluetooth devices can detect its presence.
off	Configures the module so that other Bluetooth devices can NOT detect its presence.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set visible on<CR>

ACK<CR>

>
```

Switch to Command Mode

The *switch to command mode* command instructs the EmbeddedBlue module to enter Command Mode.

Syntax

<2 second pause>*esc sequence*<2 second pause>

Parameters

esc sequence	Three consecutive instances of the escape character. The factory default escape character is the plus sign (+). A different escape character can be set by using the Set Escape Character command.

Example

Command Mode	`>con 00:0C:84:00:07:D7<CR>`
	`ACK<CR>`
Data Mode	`>This text is sent in data mode<CR>`
	`<2 second pause>+++<2 second pause><CR>`
Command Mode	`>get addr<CR>`
	`ACK<CR>`
	`00:0C:84:00:05:29<CR>`
	`>ret<CR>`
	`ACK<CR>`
Data Mode	`>This text is sent in data mode<CR>`
	`<2 second pause>+++<2 second pause><CR>`
Command Mode	`>dis<CR>`
	`ACK<CR>`
	`>`

Version

The *version* command returns the current firmware version of the EmbeddedBlue module.

Syntax

ver [all] <CR>

Parameters

all	An optional parameter used to return the build number, model number, serial number, and manufacturer.

Example

```
>ver all<CR>

ACK<CR>

Firmware Version: 2.0<CR>

Firmware Build: 247<CR>

Model Number: eb506<CR>

Serial Number: 1008<CR>

Manufacturer: A7 Engineering<CR>

>
```

This page intentionally left blank.

Error Codes

While using the eb506 you may encounter an error. Below is a listing of all eb506 error codes with a description of what causes the error to occur.

Error Code	Description
1	General connection failure.
2	Connection attempt failed. This error occurs when attempting to connect with an invalid Bluetooth address or a device that is not available.
3	Command not valid while active. This error occurs when there is an active connection and a command is issued that is not valid while connected with a remote device.
4	Command only valid while active. This error occurs when there is not an active connection and a command is issued that is only valid while connected with a remote device.
5	An unexpected request occurred. This error occurs when the remote device makes an invalid request. This is typically seen with older Bluetooth devices that may have errors in their firmware.
6	Connection attempt failed due to a timeout.
7	Connection attempt was refused by the remote device. This error typically occurs when the security settings of the remote and local device are incompatible. It can also occur when establishing a connection with security set to open if the remote and local passkeys do not match.
8	Connection attempt failed because the remote device does not support the Serial Port Profile.
9	An unexpected error occurred when deleting trusted devices.
10	Unable to add a new trusted device. This error will occur if you attempt to have more than twenty five simultaneously trusted devices.
11	Trusted device not found. This error occurs when the trusted device address is not recognized.
12	Command not valid during startup. This error occurs when a command has been issued before the EmbeddedBlue module is fully powered up and initialized.

Table 1: eb506 Error Codes

This page intentionally left blank.

Technical Specifications

Operating Parameters

The operating parameters of the eb506 are shown below in Table 2.

Transmit Power	4dBm (max) class 2 operation	
Open Field Range	eb506-AHC-IN (surface mount antenna) – 10 meters (32 feet) eb506-AHC-EN (external wired antenna) – 100 meters (328 feet) *(Actual range is dependent upon antenna selection, mounting location, and environment.)*	
Receiver Sensitivity	-85dBm	
Operating Temp.	0° to 70°C	
Supply Power	3.3VDC	
Current Consumption	115.2kbps data transfer: 35mA 38.4kbps data transfer: 30mA 9.6kbps data transfer: 25mA	connected and idle: 8mA no connection: 3mA shutdown mode: 1.5μA
Interfaces	3.3V logic level UART Baud rate 9.6k – 230.4k Flow control: RTS/CTS or none	
Connector	Two 17 pin 2mm headers compatible with RCM3000, RCM3100, RCM3200, RCM3300, and BL2500.	
Antenna	Internal surface mount or external through SMA connector	
Bluetooth Support	Version 1.2 compliant with profiles L2CAP, RFCOMM, SDP, SPP	
Firmware	Upgradeable via PC application	

Table 2: eb506 Operating Parameters

Dimensions

The dimensions of the eb506 are shown below in Table 3. Please reference Figure 14 to locate the referenced dimension on the eb506.

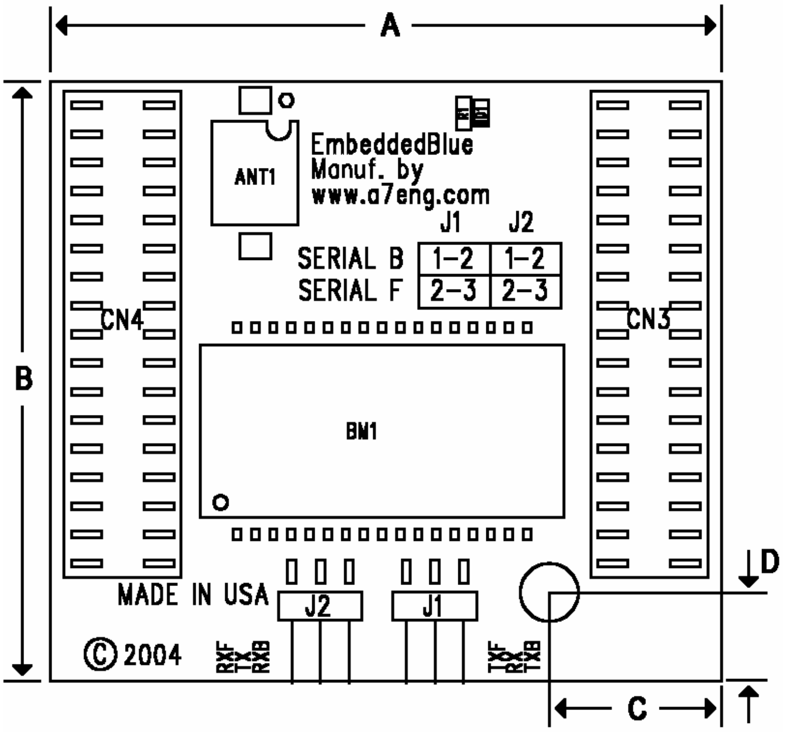


Figure 14: eb506 Dimensions

Dimension	inches	mm
A	1.85	46.99
B	1.65	41.91
C	0.47	12.06
D	0.24	6.22

Table 3: eb506 Dimensions

Pin out

The eb506 module features two 17 pin 2mm headers for direct connection with RCM3000, RCM3100, RCM3200, RCM3300, and BL2500. Currently, seven of the pins are in use (five when flow control is set to none). There are actually nine pins described below rather than seven because two different serial ports can be selected using jumpers J1 and J2. The other pins are reserved for future use.

Pin	Z-World Pin	Function	Description	Usage
CN1 – 1	GND	GND	Ground	Required
CN1 - 19	PC4	RX	Serial Receive line from eb506 if jumper J1 set on Port B	Alternate
CN1 - 20	PC5	TX	Serial Transmit line from eb506, if jumper J2 set on Port B	Alternate
CN1 - 25	PG2	RX	Serial Receive line from eb506 if jumper J1 set on Port F	Alternate
CN1 - 26	PG3	TX	Serial Transmit line from eb506, if jumper J2 set on Port F	Alternate
CN1 - 27	PD4	Status	Bluetooth connection status (0 = not connected, 1 = connected)	Required
CN1 - 28	PD5	Mode	Command/data mode toggle (0 = command, 1 = data)	Required
CN1 - 29	PD2	RTS	Request-to-Send on the serial port interface between the eb506 and the RabbitCore	Optional
CN1 - 30	PD3	CTS	Clear-to-Send on the serial port interface between the eb506 and the RabbitCore	Optional
CN2 - 13	PE7	On/Off	Powers the eb506 up or down (0=off, 1=on)	Required
CN2 - 31	VCC	VCC	Power	Required
CN2 - 32	GND	GND	Ground	Required
CN2 - 34	GND	GND	Ground	Required

Table 4: eb506 Pin out Description

This page intentionally left blank.

Frequently Asked Questions

Question: How do I obtain eMbedded Visual C++ 4.0 to develop Pocket PC applications?

Answer: The eMbedded Visual C++ 4.0 development tool is available from Microsoft. In addition, you will need eMbedded Visual C++ 4.0 SP2 and the SDK for Windows Mobile™ 2003-based Pocket PCs. These tools can be downloaded free of charge from the Microsoft Windows Mobile web site: http://www.microsoft.com/windowsmobile.

Question: Why is my eb506 not displayed when I try to discover it from my PC or Pocket PC?

Answer: Verify that the eb506 module is properly powered. It is likely you will discover the eb506 on the first attempt; however, because Bluetooth discovery is not deterministic, discovery on the first attempt is not guaranteed. On the PC or Pocket PC, use the refresh option to search for devices again. Verify that the discoverable mode setting in the eb506 is set to on.

Question: I can discover my eb506, but why am I unable to establish a connection?

Answer: Verify that the connectable mode setting in the eb506 is set to on and that security is set either to off or open. In closed security mode only devices that have already established a trusted relationship will be allowed to connect.

Question: When I try to connect from an EmbeddedBlue device with 1.0 firmware to one with 2.0 firmware the connection attempt times out and then fails with Error 2. Why?

Answer: Version 1.0 firmware did not support passkey security and trusted relationships, which is enabled as the default in version 2.0 firmware. To connect from a version 1.0 device you will need to disable security on the version 2.0 device with the “set security off” command.

Question: I am transmitting large packets of data between two ZWorld Core Modules using two eb506’s. Now and then I notice that some data seems to be lost. What is going on?

Answer: Bluetooth is a reliable point to point protocol much like TCP\IP. If transmitted data is lost or corrupted over the air it will automatically and seamlessly be retransmitted. As long as the eb506 status line tells you that there is a valid connection, you can be confident that all data will be delivered properly.

The most likely cause of this data loss involves the size of the RCM module UART buffer. The default buffer size is 31 bytes and can be overwhelmed if large amounts of data are flowing into it more quickly than they are removed. You can easily adjust this by changing the buffer size defines for the port that you are using. Refer to the Dynamic C User's Guide for more details.

Question: I used the *set visible* command to make the eb506 module not visible to other devices, but when I perform a scan from my PC I still see the device. Why?

Answer: Most of the PC Bluetooth implementations cache device scan results to save time. If you located the eb506 module before making it invisible, the PC will remember the device even though it can no longer be seen. These results are typically only cached until the Bluetooth stack is reset, so if you reboot the PC or remove and reinsert the dongle you should no longer see the device.

Contact Information

A7 Engineering provides technical support through email, an online discussion forum, and by telephone. It is recommended that you use the online forum as the first line of support so that you, and the community at large, can benefit from the postings. The forum is monitored by A7 employees to assure feedback in a timely manner.

Website: www.a7eng.com

Support Email: support@a7eng.com

Online Forum: http://www.a7eng.com/support/forum/forum.htm

Sales Email: sales@a7eng.com

A7 Engineering, Inc.

12860 C Danielson Court
Poway, CA 92064
858.679.7708 main
858.391.5616 fax